TAMPEREEN TEKNILLINEN YLIOPISTO

MARKUS KORKEE

ETÄHALLINTALAITTEEN LIITTÄMINEN OSAKSI CANOPEN-JÄRJESTELMÄÄ

Diplomityö

Tarkastaja: professori Hannu Koivisto
Tarkastaja ja aihe hyväksytty
Teknisten tieteiden tiedekuntaneuvoston
kokouksessa 8. toukokuuta 2013

TIIVISTELMÄ

TAMPEREEN TEKNILLINEN YLIOPISTO
Automaatiotekniikan koulutusohjelma
KORKEE, MARKUS: Etähallintalaitteen liittäminen osaksi CANopen-järjestelmää
Diplomityö, 56 sivua, 7 liitesivua
Kesäkuu 2013
Pääaine: Automaation ohjelmistotekniikka
Tarkastaja: professori Hannu Koivisto
Avainsanat: CANopen, etähallinta, protokollasilta, reititys

Etähallinta on tärkeä osa useita nykyaikaisia automaatiojärjestelmiä. Sen avulla mahdollistetaan järjestelmien tarkkailu ja ohjaus keskitetysti, vaikka järjestelmät sijaitsisivat maantieteellisesti kaukana toisistaan. Wapice Oy on kehittänyt oman WRM-etähallintajärjestelmän (*Wapice Remote Management*), joka mahdollistaa useiden teollisuudessa paljon käytettyjen protokollien etähallinnan.

Tämän työn tarkoituksena on tutkia, miten WRM-etähallintajärjestelmän etähallintalaite voidaan liittää osaksi CANopen-järjestelmää. Erityisesti tutkimuksen kohteena on, voidaanko etähallintalaite liittää ainoastaan passiiviseksi kuuntelijaksi vai onko myös järjestelmän etäkonfigurointi mahdollista CANopen-protokollan palveluobjektien avulla. Työn tavoitteena on löytää ja toteuttaa menetelmä, jonka avulla voidaan kuunnella väylältä haluttuja prosessisignaaleita ja hätäviestejä sekä etäkonfiguroida väylän laitteita. Ratkaisulta vaaditaan, että se ei saa sisältää tiettyyn järjestelmään sidottuja toimintoja ja etähallinnan liittäminen osaksi CANopen-järjestelmää ei saa aiheuttaa muutoksia väylän alkuperäisiin laitteisiin.

Työ jakaantuu kahteen osaan. Teoriaosassa käydään läpi CANopen-protokollan tärkeimpiä ominaisuuksia, vertaillaan CANopen-protokollaa muihin CAN-pohjaisiin ylemmän tason protokolliin sekä esitetään neljä menetelmää, joiden avulla etähallintalaite voidaan liittää osaksi CANopen-järjestelmää. Teoreettisen tarkastelun pohjalta toteutukseen valittiin CANopen-siltaratkaisu. CANopen-siltaratkaisussa etähallintalaite toimii siltana väylän hallintalaitteen ja orjalaitteiden välillä. Työn toinen osa koostuu siltaratkaisun reititykseen liittyvistä mittauksista, käytetyistä suunnitteluratkaisuista sekä lopullisen siltatoteutuksen testauksesta ja reititysviiveiden määrityksestä SAE-suorituskykytestin avulla.

Työssä esitetään CANopen-sillan suunnitteluratkaisut ja ohjelmistorakenne, joiden avulla on mahdollista saavuttaa halutut toiminnallisuudet ja täyttää menetelmälle esitetyt vaatimukset. Siltaratkaisun reititykseen kiinnitetään erityistä huomiota, koska reititys tulee optimoida siten, että viiveet eivät nouse tarkasteltavan järjestelmän kannalta liian suuriksi. Työssä esitettävien mittausten perusteella voidaan todeta, että siltaratkaisun reititys saatiin optimoitua riittävälle tasolle, joten työn tulosten perusteella CANopen-siltaratkaisu integroitiin osaksi WRM-järjestelmää.

ABSTRACT

TAMPERE UNIVERSITY OF TECHNOLOGY
Master's Degree Programme in Automation Engineering
KORKEE, MARKUS: Integrating remote management device into CANopen system
Master of Science Thesis, 56 pages, 7 Appendix pages
June 2013
Major: Automation Software Engineering
Examiner: Professor Hannu Koivisto
Keywords: CANopen, remote management, protocol bridge, routing

Remote management is important part of many modern automation systems. It allows centralized monitoring and controlling of systems even though they are geographically located far from each other. Wapice Ltd has developed WRM-system (*Wapice Remote Management*) which enables remote management of many protocols widely used in industry.

The purpose of this work is to study how WRM-system's remote management device can be integrated into CANopen-system. In particular, the research focuses on whether remote management device can only be used as passive listener or is it also possible to remotely configure system with CANopen's service data objects. The aim is to find and implement a method that can be used to listen to desired process signals and emergency messages as well as remotely configure bus' slave devices. It is also required that no system specific solutions should be used and no modifications should need to be done to the original bus in order to integrate the remote management into the system.

The work is divided into two parts. In theoretical part CANopen-protocol's most important features are presented, CANopen is being compared to other CAN-based upper level protocols and four different methods to connect remote management device into CANopen-system are introduced. Based on the theoretical study, CANopen-bridge was chosen to be implemented. In this scenario remote management device operates as bridge between the manager device and slave devices of the CANopen-bus. The second part of the work consists of routing delay measurements of the bridge, design solutions used and testing the final bridge implementation and its routing delays with SAE-benchmark.

The work presents CANopen-bridge design and software architecture that fulfill the desired functionalities and meet the requirements defined. Bridge's routing delays are studied with extra care because routing delays have to be optimized so that they do not interfere with the original bus' functionalities. Based on the measurements presented in this work it can be concluded that routing was optimized into a sufficient level and thus CANopen-bridge was integrated into WRM-system.

ALKUSANAT

Haluan kiittää työnantajaani Wapice Oy:tä erittäin mielenkiintoisesta aiheesta, opastuksesta ja laitteistosta. Näiden ansiosta diplomityöni tekeminen sujui lähes ongelmitta alusta loppuun asti ja pysyi suunnittelemassani aikataulussa erittäin hyvin. Erityiskiitos esimiehelleni ja ohjaajalleni Tommi Moisiolle sekä ohjelmistosuunnittelija Teemu Siuruaiselle, joiden neuvot ja huomiot olivat tärkeässä osassa työtä suoritettaessa. Lopuksi haluan kiittää myös työni tarkastajaa, professori Hannu Koivistoa, erittäin nopeasta työn tarkastuksesta sekä neuvoista erityisesti työn rakennetta koskevissa kysymyksissä.

Sastamalassa 20.05.2013

_____

Markus Korkee

SISÄLLYS

TERMIT JA NIIDEN MÄÄRITELMÄT

Asetustiedosto	*DCF, Defice Configuration File*, elektronisen tietolehden laitekohtainen instanssi.
CAN	*Controller Area Network*, automaatioväylä, jota käytetään erityisesti ajoneuvoissa, koneissa ja teollisuuslaitteissa.
CAN Kingdom	Kvaserin kehittämä ja ylläpitämä CAN-väylään pohjautuva ylemmän tason automaatioprotokolla.
CANopen	CiA-järjestön ylläpitämä ja kehittämä CAN-väylään pohjautuva ylemmän tason automaatioprotokolla.
CANopen-ohjelmistopino	Ohjelmistopino, joka koostuu objektikirjastosta ja CANopen-viestikehysten käsittelystä.
CiA	*CAN in Automation*, käyttäjäorganisaatio, joka kehittää ja ylläpitää useita CAN-pohjaisia ylemmän tason protokollia kuten CANopen-protokollaa.
CIP	*Communucation and Information Protocol*, automaatioteollisuudessa käytetty ylemmän tason protokolla.
DeviceNET	CAN-väylään ja CIP-protokollaan pohjautuva ylemmän tason automaatioprotokolla.
Elektroninen tietolehti	*EDS, Electronic Datasheet*, tietolehti, joka kuvaa standardoidulla tavalla laitteen toiminnallisuuden.
Esiprosessorihaaravihje	*Branch hint*, vihje, jolla voidaan ilmaista, mihin ohjelmahaaraan ohjelman suoritus todennäköisimmin päätyy.
HAL	*Hardware Abstraction Layer*, laitteiston yleistyskerros, jonka avulla piilotetaan laitteiston ajurit ja itse laitteisto ylemmiltä tasoilta.
Heartbeat-menetelmä	Protokolla, jossa CANopen-laite lähettää tietyin väliajoin omaa tilatietoaan väylälle.

Hätäviesti	*EMCY message, Emergency message*, viesti, jonka CANopen-laite lähettää väylälle virheen seurauksena.
Inline-määre	Ohjelmoinnissa käytetty määre, jonka avulla ilmaistaan, että funktiota ei kutsuta vaan kutsukohtaan kopioidaan funktion koodi.
ISOBUS	Maatalouskoneiden yhteydessä käytettä protokolla, joka pohjautuu SAE J1939 -protokollaan.
Keskeytys	Signaali, joka saa suorittimen keskeyttämään meneillään olevan ohjelman suorituksen ja siirtymään suorittamaan keskeytyskohtaisen keskeytyskäsittelijän.
Keskitin	Laite, joka yhdistää tiedonsiirtoverkon osia OSI-mallin kerroksella yksi (fyysinen kerros). Sisältää useita portteja.
Kontekstinvaihto	Prosessin tilan (kontekstin) talletus/palautus prosessin keskeytyksen seurauksena.
Kytkin	Laite, joka yhdistää tiedonsiirtoverkon osia OSI-mallin kerroksella kaksi (siirtokerros). Sisältää useita portteja.
Laiteprofiili	*Device profile*, määrittelee yksityiskohtaisesti laitetyyppikohtaiset liikennöintiparametrit ja konfigurointipalvelut.
Laitteen vartiointi	*Node Guarding*, protokolla, jossa CANopen-laitteelta kysellään aktiivisesti sen tilaa.
LLC-komponentti	*LowLevelCAN*, Wapice Oy:ssä kehitetty HAL-komponentti, joka yleistää CAN-laitteiston ja -ajurit.
Lähetysprosessisignaali	*TPDO, Transmit Process Data Object*, prosessisignaali, jonka laite lähettää väylälle.
NMEA 2000	*National Marine Electronics Association*, vedessä liikkuvien laitteiden ja koneiden yhteydessä käytettä protokolla, joka pohjautuu SAE J1939 -protokollaan.

Objektikirjasto	*Object dictionary*, CANopen-laitteen ydin, joka sisältää kaikki tiedonsiirron ja sovelluksen objektit.
Oikosiirto	*DMA, Direct Memory Access*, tiedon kopiointi tietokoneen sisällä kuljettamatta kopioitavaa tietoa suorittimen kautta.
OSI-referenssimalli	*Open Systems Interconnection Reference Model*, malli, joka kuvaa tiedonsiirtoprotokollien yhdistelmän seitsemässä kerroksessa.
Palveluobjekti	*SDO, Service Data Object*, CANopen-viestityyppi, joka mahdollistaa laitteen objektikirjaston lukemisen ja kirjoittamisen pyyntö-vastaus -mallin mukaisesti.
Palveluobjektikanava	Tiedonsiirtokanava, jonka avulla voidaan suorittaa palveluobjektioperaatioita (luku tai kirjoitus).
Palveluobjektikanavan hallintalaite	CANopen-verkon laite, joka hallitsee palveluobjektikanavia ja antaa myöntää tai kieltää luvan niiden käyttöön.
Prosessisignaali	*PDO, Processs Data Object*, CANopen-viestityyppi, jonka avulla voidaan siirtää prosessitietoa tuottaja-kuluttuja -mallin mukaisesti.
Prosessisignaalin kuvaus	*PDO mapping*, kuvaa mitä arvoja tietty prosessisignaali sisältää.
REST	*Representational State Transfer*, HTTP-protokollaan perustuva arkkitehtuurimalli ohjelmointirajapintojen toteuttamiseen.
SAE	*Society of Automotive Engineers*, yhdysvaltalainen autoalan standardointijärjestö.
SAE J1939	SAE-järjestön hallinnoima CAN-väylään pohjautuva ylemmän tason automaatioprotokolla.

SDS	*Smart Distrubuted System*, erityisesti hajautettujen binäärisensoreiden ja -toimilaitteiden kanssa käytetty ylemmän tason protokolla, joka usein pohjautuu CAN-väylään.
Silta	Kytkimen erikoistapaus, joka sisältää ainoastaan kaksi porttia.
Singleton-suunnittelumalli	Suunnittelumalli, jossa komponentista tai oliosta luodaan ainoastaan yksi instanssi.
Tahdistusviesti	CANopen-viesti, johon voidaan sitoa toimintoja, joiden täytyy tapahtua mahdollisimman samanaikaisesti.
Tarkkailija-suunnittelumalli	*Observer Design Pattern*, suunnittelumalli, jossa rekisteröidytään tietyn kohteen tarkkailijaksi.
Tietoverkon hallintaisäntä	*NMT Master, Network Management Master*, laite, joka suorittaa CANopen-verkon hallinnan.
Toistin	Keskittimen erikoistapaus, joka sisältää ainoastaan kaksi porttia.
Topologia	Kuvaa verkon perusrakenteen.
Vastaanottoprosessisignaali	*RPDO, Receive Process Data Object*, prosessisignaali, jonka laite vastaanottaa väylältä.
WCC-ajuri	*Wapice Custom CAN*, Wapice Oy:ssä kehitetty CAN-ajuri.
WRM-järjestelmä	*Wapice Remote Management*, Wapice Oy:ssä kehitetty etähallintajärjestelmä.

1 JOHDANTO

Etähallinta on tärkeä osa useita nykyaikaisia automaatiojärjestelmiä. Sen avulla mahdollistetaan järjestelmien tarkkailu ja ohjaus keskitetysti, vaikka järjestelmät sijaitsisivat maantieteellisesti kaukana toisistaan. Usein automaatiojärjestelmiä asennetaan niin hankaliin paikkoihin, että etähallinta on ainoa vaihtoehto. Etähallinnan avulla on mahdollista kerätä järjestelmästä tärkeää tietoa, jota voidaan käyttää hyväksi esimerkiksi huoltoja ennakoitaessa. Lisäksi järjestelmän asetusten muuttaminen on usein mahdollista etähallintalaitteen avulla. Näin ollen etähallinnan avulla voidaan lisätä järjestelmän kustannustehokkuutta.

Wapice Oy on Vaasassa perustettu teollisuusyritysten ohjelmistoratkaisuihin ja tietojärjestelmien integrointiin erikoistunut yritys. Wapicella on kehitetty oma WRM-etähallintajärjestelmä (*Wapice Remote Management*), jonka tarkoituksena tarjota asiakkaalle täysipainoinen etähallintaratkaisu: elektroniikka, palvelin sekä ohjelmisto. WRM-järjestelmän avulla on pystyttävä liittymään osaksi yleisimpiä teollisuuden väyliä ja sen tulee tukea kyseisiin väyliin liittyviä ylemmän tason protokollia. Näihin protokolliin lukeutuu myös tässä työssä käsiteltävä CAN-väylän ylemmän tason protokolla CANopen. CANopen on etenkin teollisuusratkaisuissa paljon käytetty protokolla, joka mahdollistaa hajautettujen automaatiojärjestelmien suunnittelun, käyttöönoton ja ylläpidon.

Tämän työn tarkoituksena on tutkia, miten CANopen-järjestelmän osaksi voidaan liittää etähallinta. Erityisesti tutkimuksen aiheena on, voidaanko etähallintalaite liittää osaksi CANopen-verkkoa ainoastaan passiiviseksi kuuntelijaksi vai voidaanko mahdollistaa myös järjestelmän etäkonfigurointi CANopen-protokollan palveluobjektien avulla. Tutkimuksen ja teoreettisen tarkastelun pohjalta on tarkoitus määrittää Wapice Oy:n WRM-järjestelmän etähallintalaitteelle vaatimukset täyttävä ohjelmistorakenne ja toteuttaa etähallintalaitteeseen CANopen-protokolla tuki määriteltyjen vaatimusten ja suoritettujen mittausten edellyttämällä tavalla. Lopullisen sovelluksen tavoitteena on, että etähallintalaitteen avulla voidaan seurata väylän liikennettä ja konfiguroida väylän laitteita.

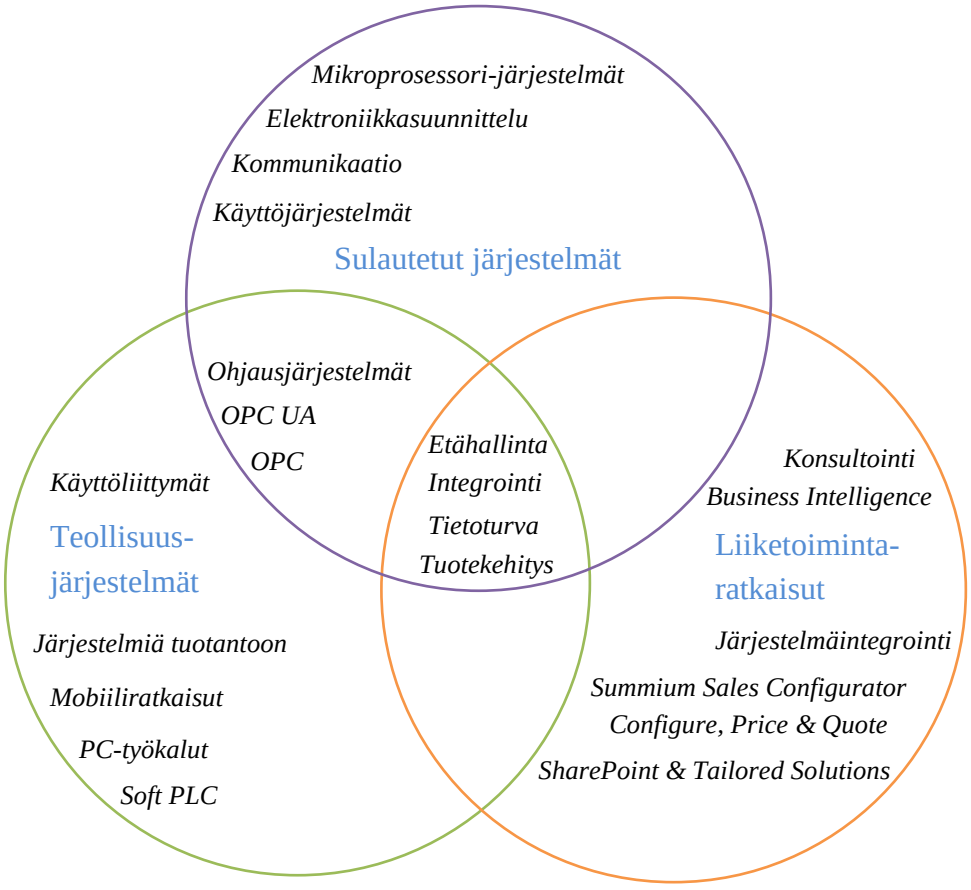
Luvussa kaksi lähdetään liikkeelle esittelemällä käytetty WRM-etähallintajärjestelmä, minkä jälkeen luvussa kolme siirrytään tarkastelemaan CANopen-protokollan teoriaa. CANopen-protokollan teorian jälkeen luvussa neljä tutkitaan muita CAN-pohjaisia ylemmän tason protokollia ja vertaillaan niitä CANopen-protokollaan. Luvussa viisi tarkastellaan tapoja, joiden avulla on mahdollista liittää etähallintalaite osaksi CANopen-järjestelmää ja tämän pohjalta valitaan toteutettava menetelmä. Lu-

vussa kuusi esitetään mittaustuloksia, joiden avulla voidaan perustella luvussa seitsemän esitetyt suunnitteluratkaisut. Luku kahdeksan esittää suunnitteluratkaisujen pohjalta tehdyn lopullisen toteutuksen tunnuslukuja. Lopuksi kaikki käsitellyt asiat vedetään yhteen luvussa yhdeksän.

2 WAPICE OY JA WRM-JÄRJESTELMÄ

Wapice Oy on itsenäinen informaatioteknologian palveluyritys, joka perustettiin Vaasassa vuonna 1999. Yhtiö on yksityisesti omistettu, enemmistöomistus on johdolla ja työntekijöillä. Ensisijaisena tavoitteena yrityksellä on vastata teollisuusyritysten ohjelmistotarpeisiin, minkä seurauksena Wapice Oy:n asiakkaisiin kuuluu yli 200 Suomen suurimpiin lukeutuvaa teollisuusyritystä. [1.]

Wapice Oy työllistää yli 240 ohjelmistoalan asiantuntijaa (tilanne vuoden 2013 toukokuussa). Yhtiön pääpaikka on Vaasassa ja muut yksiköt ovat sijoittuneet Tampereelle, Ouluun, Seinäjoelle ja Hyvinkäälle. [1.] Wapice Oy koostuu kolmesta segmentistä, jotka on esitetty alla olevassa kuvassa 2.1.

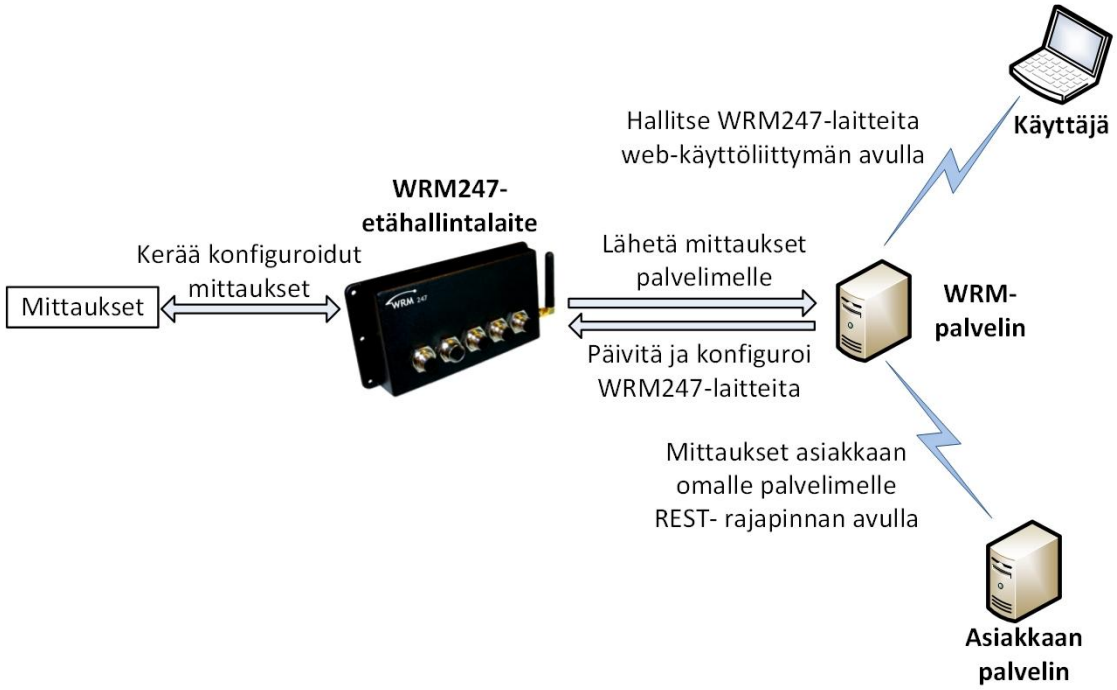


Kuva 2.1. *Wapice Oy:n segmentit [1].*

Kuten kuvasta huomataan, ovat kaikki segmentit painottuneet vahvasti teollisuuden tarpeisiin. Tämän diplomityön kannalta tärkein osa-alue eli etähallinta löytyy kaikkien

kolmen segmentin leikkauksesta. Eräs tärkeä etähallinnan tehtävä onkin tarjota tietoa sulautetuista järjestelmistä aina teollisuusjärjestelmien- ja liiketoiminnantasoille.

Wapice Oy:n oma etähallintatuote on WRM-järjestelmä (*Wapice Remote Management).* Sen konsepti on esitetty alla olevassa kuvassa 2.2.



Kuva 2.2. *WRM-järjestelmän konsepti.*

Idea on tarjota asiakkaalle kokonainen etähallintajärjestelmä: elektroniikka, ohjelmistot sekä palvelin. WRM-järjestelmässä on kaksi erityisen tärkeää roolia: WRM247-tiedonkeruulaite sekä WRM-palvelin. Seuraavat luvut esittelevät tarkemmin kyseiset roolit.

2.1 WRM247-etähallintalaite

WRM247-etähallintalaitteen laitteisto ja ohjelmisto on kokonaan suunniteltu ja toteutettu Wapice Oy:n toimesta. Laitteen on tarkoitus toimia edullisena etähallinta- ja tiedonkeruulaitteena, joka tarjoaa paljon liityntöjä teollisuuden tarpeisiin sekä tuen useimmille paljon käytetyille protokollille. WRM247-laite on esitetty kuvassa 2.3.

Kuva 2.3. *WRM247-laite [2].*

Laitteen saa useilla eri varustetasoilla, jolloin kuvassa vasemmalta lähtien numeroitujen M12-standardin mukaisien liitinpaikkojen määrä vaihtelee. Kuvassa on esitetty täyskalustettu malli. Taulukko 2.1 esittää kunkin liittimen sisältämät liitynnät.

Taulukko 2.1. *WRM247-laitteen M12-liittimien liitynnät [2].*

Liitin 1	**Liitin 2**	**Liitin 3**	**Liitin 4**	**Liitin 5**
Ethernet	Käyttöjännite	1-wire	USB	CAN-väylä 2
Debug RS-232	DIN 1, 2	AIN 2	RS-485	DOUT 2, 3, 4
5 V ulostulo	DOUT jännite sisään	Virtasilmukka (20 mA)	AIN 1	RS-232
	DOUT 1	DIN 3,4		
	CAN-väylä 1			

Taulukossa DIN tarkoittaa digitaalista sisääntuloa, DOUT digitaalista ulosmenoa ja AIN analogista sisääntuloa. DOUT jännite sisään tarkoittaa jännitettä, joka syötetään digitaaliseen ulosmenoon, kun niihin kirjoitetaan looginen arvo yksi. Liitin kaksi on aina pakollinen, koska se sisältää laitteen käyttöjännitteen, jonka alue on 9-34 volttia. Yhteysvaihtoehtoina ovat Ethernet, GPRS tai USB-liityntään liitettävä 3G/4G-adapteri, joka voidaan sijoittaa myös kotelon sisään. Lisäksi laitteesta on sisäänrakennettu GPS sekä kiihtyvyysanturi. Tämän työn kannalta tärkeimmät liitynnät ovat CAN-liitynnät, joita WRM247-laitteesta löytyy kaksi kappaletta. Laitteessa on ARM9-pohjainen prosessori, joka toimii 200 MHz kellotaajuudella.

WRM247-laitteen perusta on Linux-käyttöjärjestelmä, jonka päälle on rakennettu Wapicen oma sovellusohjelmisto. Kyseinen ohjelmisto koostuu kahdesta pääosasta: sovelluslogiikasta sekä palvelinkommunikaatiosta. Sovelluslogiikka koostuu ohjelmis-

tomoduuleista, joista jokainen on vastuussa tietystä protokollasta tai laitteiston osasta. Erillisiä moduuleita ovat esimerkiksi analogia- ja digitaalitulojen ja -lähtöjen hallinta sekä Modbus-protokollamoduuli. Tämän diplomityön tavoitteena on kehittää CANopen-protokollamoduuli, joka mahdollistaa WRM247-laitteen käytön osana CANopen-järjestelmää.

2.2 WRM-palvelin

WRM-palvelin on monipuolinen työkalu WRM247-laitteiden hallintaan ja mittausten tarkasteluun. WRM-palvelin tarjoaa käyttäjille web-käyttöliittymän, jonka avulla toiminnot voidaan suorittaa. Jokaiselle asiakkaalle luodaan oma virtuaalinen tuotantopalvelin, jonne asiakkaalle luodaan haluttu määrä käyttäjätunnuksia halutuilla oikeustasoilla. WRM-palvelimen avulla käyttäjä voi [2]:

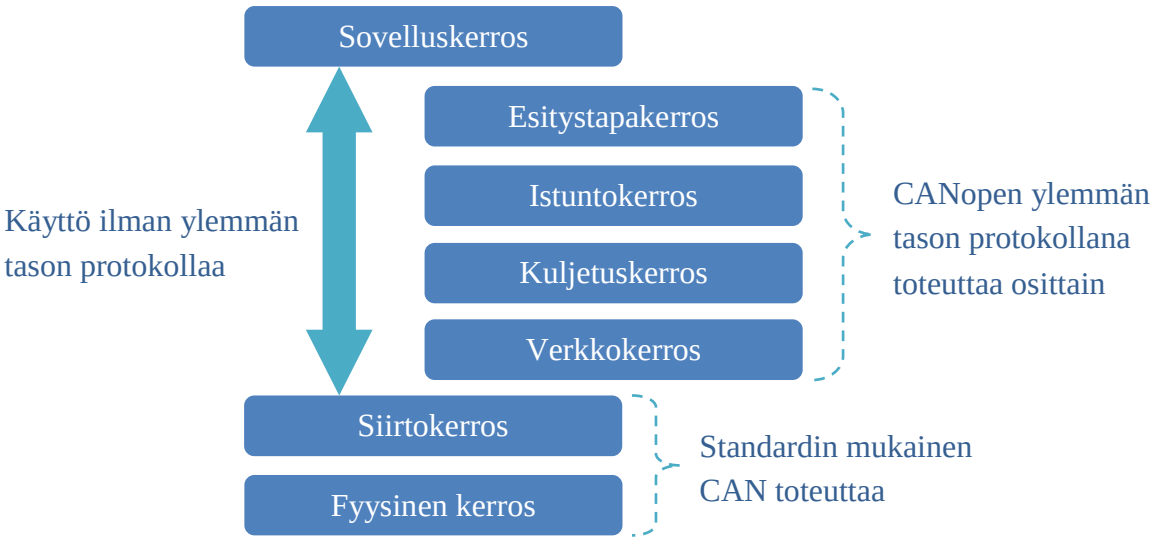
1. Lisätä uusia WRM247-laitteita
2. Päivittää laitteen asetuksia, kuten:
    - Halutut mittaukset
    - Mittauksien tallennusaikaväli
    - Ladattavat ohjelmistomoduulit
3. Päivittää laitteen ohjelmiston
    - Uuden ohjelmistoversion mukana tulee uusia ominaisuuksia ja ohjelmistomoduuleita
4. Piirtää kuvaajia kerätyistä mittauksista
    - Palvelin tarjoaa työkalut kuvaajien piirtoon halutulta aikaväliltä
5. Tallentaa mittauksia esimerkiksi csv-tiedostoksi tarkempaa käsittelyä varten

Palvelin tarjoaa myös REST-rajapinnan (*Representational State Transfer*), jonka avulla on helppo ladata mittaukset asiakkaiden omille palvelimille ja integroida WRM-järjestemä osaksi muita järjestelmiä. Kyseisen rajapinnan kautta voidaan myös ohjata WRM-laitteita.

3 CANOPEN-PROTOKOLLAN TEORIA

CAN-väylä (*Controller Area Network*) kehitettiin alun perin autoteollisuuden tarpeita varten. CAN-väylän historian katsotaan alkaneen vuodesta 1983, jolloin ajoneuvoteollisuuden elektroniikkavalmistaja Bosch alkoi kehittää kyseistä väylää ajoneuvon sisäistä tietoverkkoa varten. CAN-väylä levisi nopeasti teollisuuden keskuudessa ja nykyään sitä käytetään erityisesti ympäristöissä, joissa virheensietokyky on tärkeää. Suosion kasvaessa ja sovellusalueiden monimutkaistuessa oli selvää, että CAN-väylä ei yksin riitä. Niinpä vuoden 1992 alussa perustettiin CAN in Automation (CiA) -järjestö, jonka tehtävä oli edistää ja kehittää CAN-väylän ylemmän tason protokollia. CiA-järjestö julkaisi vuonna 1995 ensimmäisen version CANopen-protokollan sovellustasosta. [3.] CANopen kehitettiin alun perin sulautettuihin automaatioverkkoihin, joissa tarvitaan joustavia konfigurointi mahdollisuuksia. Erityisesti se tarkoitettiin käytettäväksi koneenohjauksessa. CANopen kasvatti suosiotaan tasaisesti ja nykyään sitä käytetään useilla sovellusalueilla kuten robotiikassa, lääketeollisuudessa, maastoautoissa, rautatieliikenteessä, merielektroniikassa, rakennus- ja sähköautomaatiossa [3]. CiA-järjestö ja sen jäsenyritykset kehittävät edellään aktiivisesti CANopen-protokollaa ja uusia standardeja julkaistaan joka vuosi. CANopen-protokollan lisäksi 1990-luvulla julkaistiin useita muitakin CAN-väylään pohjautuvia ylemmän tason protokollia. Näihin lukeutuvat muun muassa tehdasautomaatiossa käytetty ODVA-järjestön (*Open DeviceNET Vendor Association*) ylläpitämä DeviceNET-protokolla, Kvaserin ylläpitämä CAN Kingdom, Honeywellin kehittämä Smart Distributed System (SDS) sekä SAE-järjestön (*Society of Automative Engineers*) kehittämä ja ylläpitämä SAE J1939. Kyseisiä protokollia ja niiden eroja CANopen-protokollaan tutkitaan luvussa neljä.

OSI-referenssimalli (*OSI Reference Model, Open Systems Interconnection Reference Model*) kuvailee seitsemän eri tasoa tietoverkkokommunikaatiolle. Se kuvailee tasot aina fyysiseltä kerrokselta sovellukseen asti. Monet alemman tason kommunikaatioteknologiat toteuttavat OSI-mallista kaksi alinta kerrosta, kuten esimerkiksi CAN-protokolla, johon CANopen-protokolla pohjautuu. CANopen-protokolla määrittelee osittain viisi ylempää OSI-mallin kerrosta, mutta ei toteuta kaikkia ylemmille kerroksille kuuluvia toiminnallisuuksia, koska niitä ei tarvita CANopen-verkoissa. Kuva 3.1 esittää, miten CAN- ja CANopen-protokollat sijoittuvat OSI-referenssimalliin.



Kuva 3.1. *CAN- ja CANopen-protokollien sijoittuminen OSI-referenssimalliin [4, s. 18].*

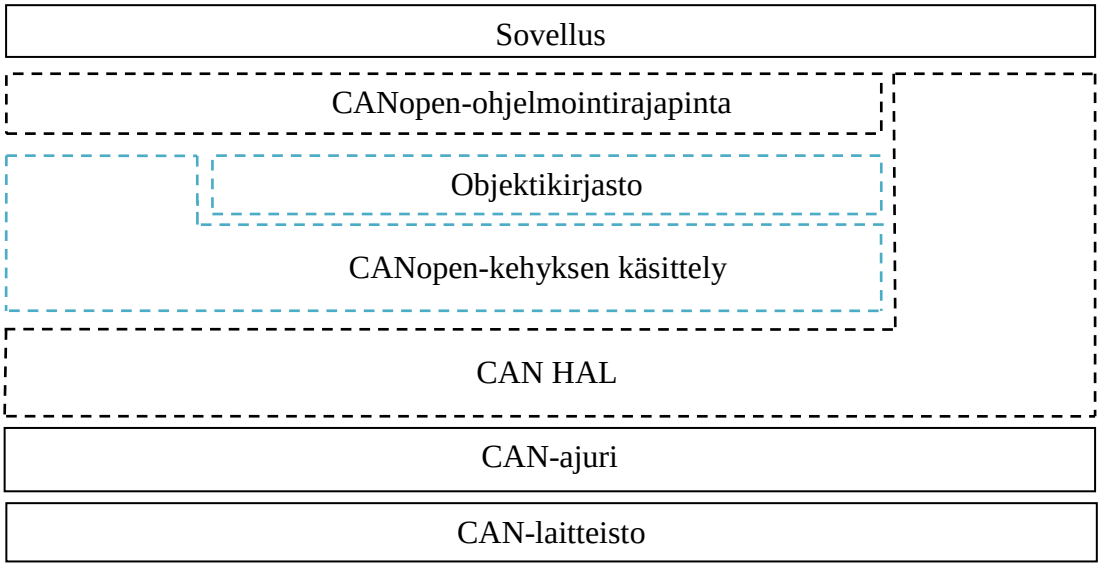
Kuvasta huomataan, että CAN-teknologiaa voidaan käyttää myös ilman ylemmän tason protokollaa, mutta ylemmän tason protokollat tarjoavat joustavuutta ja huomattavasti lisäominaisuuksia sovelluskerrokselle. Taulukossa 3.1 on kuvattu, mitä CANopen-protokollan toiminnallisuuksia ja ominaisuuksia eri OSI-mallin tasoille kuuluu.

Taulukko 3.1. *CANopen-protokollan ominaisuudet suhteessa OSI-malliin [4, s 19–21].*

Kerros	**CANopen-ominaisuus**
Sovellus	• Sovellukset, jotka käyttävät hyväksi CANopen-protokollaa ohjelmistopinon avulla
Esitys	• CANopen tarjoaa standardoidun tiedon esitystavan objektikirjaston avulla
Istunto	• Valtuuden hallinta palveluobjektikanavien avulla
Kuljetus	• Palveluobjektiviestien automaattisen osituksen avulla voidaan siirtää suurempia tietomääriä kuin mitä yhteen tietokehykseen mahtuu
Verkko	• CANopen palveluobjektikanavat kohteen osoitukseen • Verkon konfigurointi palveluobjektien avulla
Siirto	CAN-protokolla tarjoaa CANopen-protokollalle: • tietokehykset • tarkistussummien laskennan ja tarkastelun • onnistuneen kehyksen siirron varmistamisen
Fyysinen	• CANopen määrittelee käytettäväksi CAN-standardin mukaisen CAN-väylän ja siihen liittyvät ominaisuudet (kuten bittien generointi ja tahdistuminen)

Kuten taulukosta huomataan, CANopen-protokolla ei toteuta kaikkia ominaisuuksia, mitä eri kerroksille on määritelty, koska niitä ei tarvita CANopen-verkoissa. Taulukossa esitellyt ominaisuudet esitellään tarkemmin tulevissa luvuissa.

CANopen-laitteiden ohjelmistorakenne on usein alla olevan kuvan 3.2 mukainen.



Kuva 3.2. *CANopen-laitteen ohjelmiston yleinen rakenne [5, s. 6].*

Kuvassa mustalla katkoviivalla esitetyt elementit näkyvät sovellusohjelmalle ja se voi käyttää niitä suoraan. CAN-laitteiston yleistyskerros (*CAN HAL, CAN Hardware Abstraction Layer*) piilottaa ylemmiltä tasoilta CAN-ajurin ja CAN-laitteiston. Kuvassa sinisellä katkoviivalla esitettyjä objektikirjastoa ja CANopen-kehyksen käsittelyä kutsutaan yhteisnimellä CANopen-ohjelmistopino. CANopen-ohjelmistopinosta on useita eri toteutuksia ja niiden käyttöä helpotetaan toteuttamalla CANopen-ohjelmointirajapinta, joka piilottaa varsinaiselta sovellukselta ohjelmistopinon. Näin CANopen-ohjelmistopinon toteutus voidaan vaihtaa ilman, että itse sovellusta tarvitsee muuttaa. Rajapintaan voidaan toteuttaa ainoastaan sellaiset toiminnot, joita laite todellisuudessa tarvitsee.

Tässä työssä käytetään väylään liitetyistä laitteista termejä hallintalaite sekä orjalaite. Hallintalaitteella tarkoitetaan CANopen-verkon laitetta, joka on hallitsee orjalaitteiden tiloja, käynnistää verkon, omistaa palveluobjektikanavat sekä hallitsee orjalaitteiden asetuksia. Hallintalaitteella on yleensä myös muita tärkeitä tehtäviä kuten esimerkiksi tahdistusviestien tuottaminen. Hallintalaite voi myös itse tuottaa prosessitietoa väylälle.

3.1 Objektikirjasto

Jokaisen CANopen-laitteen ydin on objektikirjasto. Se voidaan ajatella hakutauluna, joka koostuu 16-bittisistä indekseistä, joista jokainen sisältää 8-bittisiä ali-indeksejä. Yhdellä indeksillä voi siis olla maksimissaan 256 ali-indeksiä. Ali-indeksi edustaa muuttujaa, joka voi olla tyypiltään ja pituudeltaan mitä tahansa. [6, s. 89.] Seuraavassa listauksessa on määritelty ominaisuudet, jotka indeksille ja ali-indeksille voidaan asettaa [5, s. 7]:

- Oikeudet (luku, kirjoitus, luku ja kirjoitus)
- Tietotyyppi
- Suurin sallittu arvo
- Pienen sallittu arvo
- Oletusarvo
- Voidaanko päivittää prosessisignaalin avulla

Jokaisen CANopen-laitteen on toteutettava oma objektikirjastonsa. Objektikirjasto sisältää kaiken tiedon, mitä voidaan tietoverkon välityksellä laitteesta lukea ja laitteeseen kirjoittaa. Se on laitteen keskitetty tietovarasto, jossa sijaitsevat sekä parametrit että signaalit. Parametreihin lukeutuu sekä liikennöinti- että sovellusparametrit. Signaalien arvot päivitetään CANopen-laitteen ja väylän välillä aina objektikirjaston kautta.

Objektikirjasto jaetaan useisiin indeksialueisiin. Alla oleva taulukko 3.2 esittää tärkeimmät CANopen-protokollan indeksialueet.

Taulukko 3.2. *Objektikirjaston indeksialueet [5, s. 7].*

Indeksialue (hex)	**Sisältö**
0x000	Ei käytössä
0x0001-0x0FFF	Tietotyypit
0x1000-0x1FFF	Laitekohtaiset tunnisteet ja liikennöintiparametrit
0x2000-0x5FFF	Valmistaja- tai sovelluskohtainen alue
0x6000-0x9FFF	Laiteprofiilit

Yllä olevassa taulukossa erityisen tärkeä alue on tietotyypeille varattu alue. Kyseinen alue sisältää sekä standardi tietotyyppejä että valmistajaspesifisiä tietotyyppejä. Seuraavana alueena ovat laitekohtaiset tunnisteet ja liikennöintiparametrit. Tämä alue sisältää tiedot CANopen-laitteen tunnistamiseen sekä väyläliikennöinnin ohjaamiseen. Valmistaja- tai sovelluskohtainen alue sisältää laitteen valmistajan määrittelemiä parametreja ja signaaleja Laiteprofiilit-alue sisältää laitteen toteuttamien laiteprofiilien parametrit ja signaalit.

Objektikirjasto sisältää kaiken tiedon, joka tarvitaan kommunikaatioon. Taulukko 3.3 esittää ne objektikirjaston alkiot, jotka jokaisen CANopen-yhteensopivan laitteen tulee toteuttaa.

Taulukko 3.3. *Objektikirjaston pakolliset alkiot [4, s. 30].*

Indeksi (hex)	**Sisältö**
0x1000	Laitteen tyyppitieto
0x1001	Virherekisteri
0x1017	Heartbeat-aika
0x1018	Tunnisteobjekti

Taulukon 3.3 alkioiden avulla CANopen-hallintalaite voi selvittää orjalaitteesta hyvin paljon oleellista tietoa. Laitteen tyyppitieto kertoo, mitä laiteprofiilia laite noudattaa. Laite voi myös olla rakennettu siten, että se ei noudata mitään laiteprofiilia. Virherekisteri kertoo laitteen mahdollisesta virhetilasta ja sen arvo lähetetään väylälle virheen sattuessa. Heartbeat-aika tarkoittaa aikaa, jonka välein laite lähettää heartbeat-viestin väylälle. Heartbeat-menetelmä on esitelty tarkemmin luvussa 3.6.1. Tunnisteobjekti sisältää neljä tärkeää ali-indeksiä: laitteen valmistajan, tuotenumeron, versionumeron sekä tuotteen sarjanumeron.

Objektikirjasto ei ota kantaa, miten voidaan tietää, mitä tietty objektikirjaston alkio sisältää. Objektikirjasto on erittäin laaja kokonaisuus, joten CANopen-hallintalaite ei voi käydä jokaisen CANopen-verkon laitteen objektikirjaston kaikkia indeksejä ja ali-indeksejä läpi ja näin tutkia, onko tietty alkio toteutettu. Niinpä hallintalaitteella tulee olla menetelmä, jonka avulla se tietää laitteiden objektikirjastojen sisällön. Nämä menetelmät on esitelty seuraavissa kahdessa luvussa.

3.2 Laiteprofiilit

Laiteprofiilit (*DF, Device Profile*) määrittelevät pakollisten objektikirjaston alkioiden lisäksi objektikirjaston alkioita, jotka laiteprofiilin toteuttavan laitteen tulee sisältää. Laiteprofiileja on olemassa esimerkiksi yleisille I/O-moduuleille sekä enkoodereille. Laite voi noudattaa maksimissaan kahdeksaa eri laiteprofiilia. [4, s. 30–31.] CANopen-hallintalaitteella on tiedossaan eri laiteprofiilien sisältämät objektikirjastojen alkiot.

CANopen-hallintalaite voi kysyä jokaiselta verkon laitteelta, mitä laiteprofiileja laite noudattaa. Kyseinen tieto on jokaisen CANopen-yhteensopivan laitteen pakko toteuttaa objektikirjaston indeksissä 0x1000. Kysely hoidetaan palveluobjektien avulla. Laiteprofiilikyselyn jälkeen hallintalaite tietää, mitä objektikirjaston alkioita tietty laite laiteprofiilin puitteissa tukee.

Laitteet tarvitsevat usein muitakin objektikirjaston alkioita kuin ne, joita laiteprofiilit määrittelevät. Alkiot ovat tällöin laitevalmistajakohtaisia, joten niitä ei voida esittää yleiskäyttöisessä laiteprofiilissa. CANopen-protokollassa tämä ei ole ongelma, vaan valmistajakohtaiset objektikirjaston alkiot tulee kuvata elektronisten tietolehtien avulla.

3.3 Elektroninen tietolehti

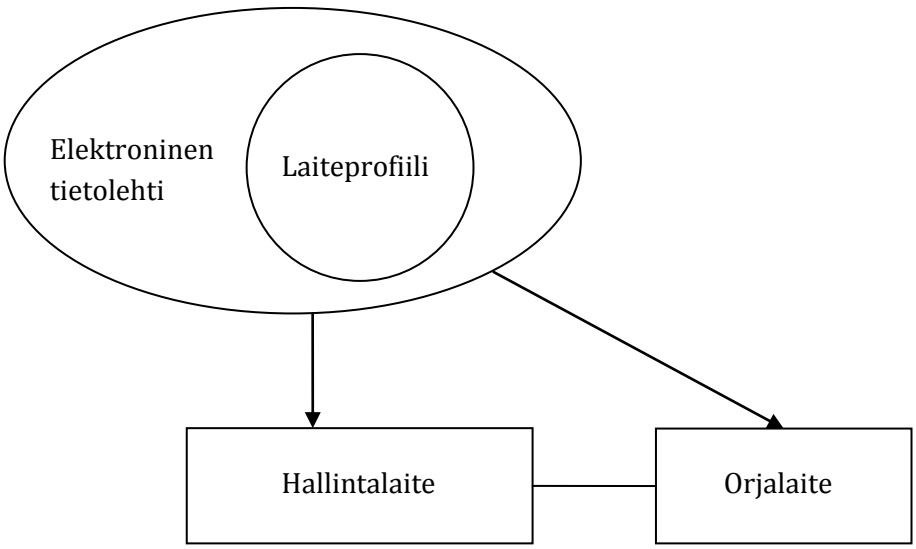
Elektroninen tietolehti (*EDS, Electronic Datasheet*) on standardoitu tapa, jonka avulla esitetään CANopen-laitteen sisältämät objektikirjaston alkiot [7, s. 6]. Elektroninen tietolehti tulee aina toimittaa laitteen mukana. Kuvassa 3.3 on esimerkki objektikirjaston yhden alkion määrityksestä elektronisessa tietolehdessä.

```
/
[1000]
ParameterName=DeviceType
ObjectType=0x07
DataType=0x0007
AccessType=ro
DefaultValue=0x00030191
PDOMapping=0
/
```

Kuva 3.3. *Objektikirjaston alkio elektronisessa tietolehdessä [4, s. 31]*.

Kuvassa on esitetty laitteen tyyppitiedon määrittely, joka on pakollinen jokaiselle CANopen-laitteelle. Määrittelystä löytyy esimerkiksi alkion tietotyyppi ja oikeusmääritykset.

Kun orjalaite havaitaan väylällä, hallintalaite etsii omasta elektronisten tietolehtien kokoelmastaan liitetyn orjalaitteen elektronisen tietolehden. Näin hallintalaite tietää suoraan käytettävissä olevat objektikirjaston alkiot. [4, s. 32.] Kuva 3.4 esittää yhteyden laitteiden, elektronisten tietolehtien ja laiteprofiilien välillä.



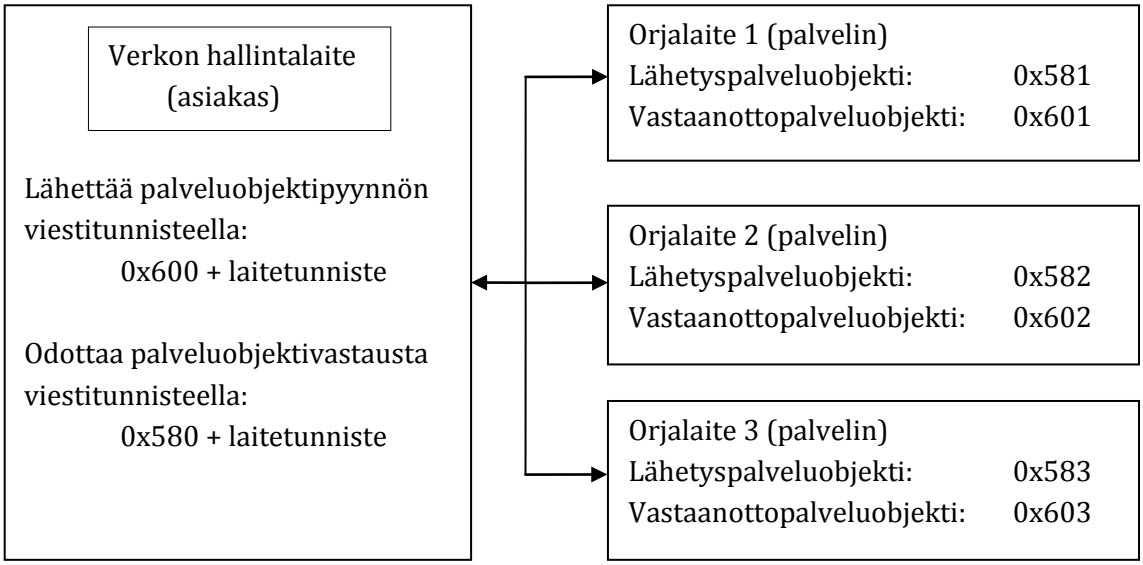
Kuva 3.4. *Elektronisten tietolehtien ja laiteprofiilien yhteys laitteisiin.*

Laite voi siis toteuttaa tietyn laiteprofiilin, mutta sen lisäksi sillä voi olla elektronisessa tietolehdessä valmistajaspesifisiä objektikirjaston alkioita. Sekä hallintalaite että orjalaite ovat tietoisia sekä laiteprofiilista että elektronisesta tietolehdestä.

Elektronisen tietolehden laitekohtainen instanssi on laitteen asetustiedosto (*DCF, Device Configuration File*). Elektronisen tietolehden idea on olla yleiskäyttöinen kuvaus usealle laitteelle. Laitteen asetustiedoston tarkoitus on tallettaa tietyn laitteen objektikirjasto. Asetustiedostossa on tallennettuna tietyt asetukset ja objektikirjaston alkioiden nykyiset arvot tai arvot, jotka objektikirjaston alkioissa tulisi olla. [7, s 18–20.] Idea on, että hallintalaite voi elektronisen tietolehden avulla selvittää, mitkä objektikirjaston alkiot laitteessa on toteutettuna ja asetustiedoston avulla voidaan tallettaa (tai palauttaa) tietyn laitteen objektikirjasto ja sen sisältämät arvot.

3.4 Palveluobjektit

Palveluobjektit (*SDO, Service Data Object*) tarjoavat hallintalaitteelle mahdollisuuden lukea ja kirjoittaa kaikkien CANopen-verkon laitteiden objektikirjastojen alkioita. Palveluobjektien avulla kommunikointi on pyyntö-vastaus -mallin mukaista. [6, s. 46.] Periaate on esitetty kuvassa 3.5.



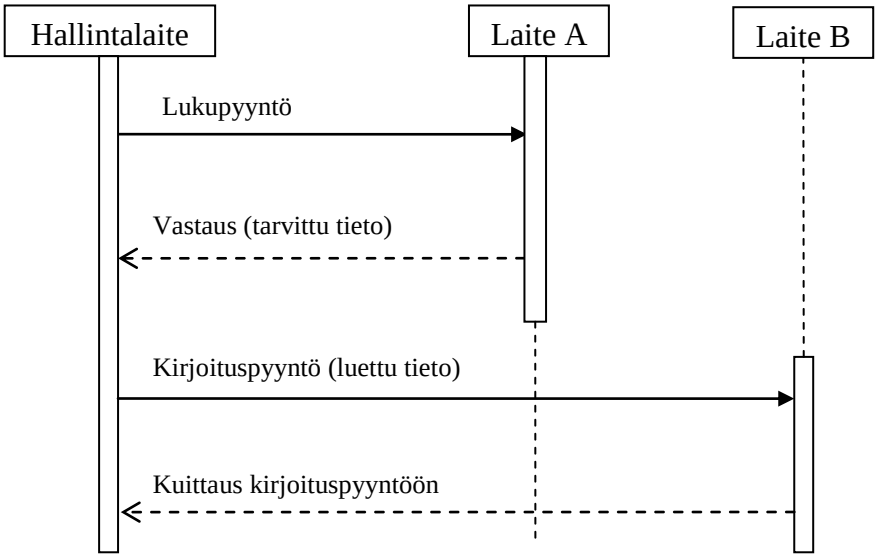
Kuva 3.5. *Palveluobjektien kanssa käytetyt viestitunnisteet [4, s. 62]*.

Kuvassa hallintalaite toimii asiakkaana, koska se pyytää tietoa palvelimena toimivalta orjalaitteelta. Hallintalaite lähettää pyynnön, jonka viestitunniste on kantaosoitteen 0x600 ja halutun laitteen laitetunnisteen summa. Laitteita voi olla maksimissaan 127, joten viestitunnisteet 0x601-0x67F on varattu kommunikaatioon yhdeltä asiakkaalta 127 palvelimelle. Vastaus puolestaan on kantaosoitteen 0x580 ja osoitetun laitteen laitetunnisteen summa. Viestitunnisteet 0x581-0x5FF on varattu 127 kanavalle palvelimelta takaisin asiakkaalle. Kokonaisuudessaan palveluobjekteille on varattu viestitunnisteavaruudesta alue 0x580-0x67F.

CANopen-verkossa voi oletuksena olla ainoastaan yksi laite, joka voi aloittaa palveluobjektikommunikaation eli toimia palveluobjektiasiakkaana. Kyseinen laite on käytännössä aina verkon hallintalaite. [5, s. 8.] Hallintalaite omistaa palveluobjektikanavat, joita sillä on yksi jokaista CANopen-verkon laitetta varten. Vain palveluobjek-

tikanavan omistava laite voi lähettää luku- tai kirjoituspyynnön, johon pyynnön vastaanottavan laitteen on vastattava palveluobjektivastauksella.

Palveluobjektien avulla voidaan toteuttaa koko tietoverkko, koska myös prosessitietoa voidaan lukea palveluobjektien avulla. Tällöin kuitenkin kaikki tieto kulkisi aina keskitetysti hallintalaitteen kautta. [4, s. 33.] Oletetaan seuraava yksinkertainen tilanne: laite B tarvitsee prosessitietoa laitteen A objektikirjastosta ja kumpikaan laitteista ei ole hallintalaite. Kuva 3.6 havainnollistaa tarvittavat operaatiot.



Kuva 3.6. *Kahden laitteen välinen tiedonsiirto palveluobjektien avulla.*

Hallintalaite lukee tarvitun tiedon laitteelta A ja kirjoittaa tämän tiedon laitteelle B. Kuten kuvasta huomataan, tämä yksinkertainen tilanne vaatii yhteensä neljä viestiä, vaikka koko tilanteen voisi korvata yhdellä viestillä suoraan laitteelta A laitteelle B. Tästä voidaan päätellä neljä huonoa puolta, kun käytetään palveluobjekteja prosessitiedon välittämiseen [4, s. 33–34]:

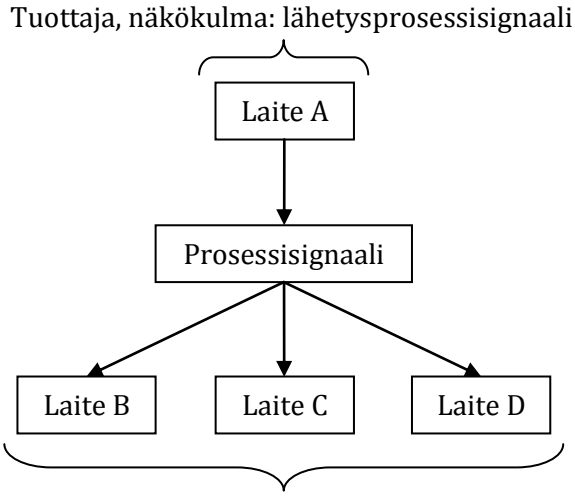
1. Hallintalaitteen tulee hoitaa kaikki liikenteen käsittely laitteiden välillä ja aktiivisesti kysellä laitteilta objektikirjastojen alkioita, joten hallintalaite kuormittuu.
2. Kaistanleveyttä käytetään turhaan, koska yksinkertaisetkin operaatiot vaativat useita viestejä väylällä.
3. Palveluobjektiviestien tietopituus on aina maksimi kahdeksan tavua, vaikka kaikkia tietotavuja ei tarvittaisikaan.
4. Viiveet kasvavat huomattavasti.

Näin ollen palveluobjekteja ei tulisi käyttää prosessitiedon siirtämiseen, koska se on tehotonta. Palveluobjektit ovat kuitenkin välttämättömiä, koska niiden avulla voidaan lukea ja kirjoittaa laitteiden liikennöinti- ja asetusparametreja.

3.5 Prosessisignaalit

Palveluobjektit eivät tarjoa tarpeeksi tehokasta tiedonsiirtoa prosessitiedolle, koska kaistanleveyttä käytetään turhaan ja viiveet kasvavat. Lisäksi palveluobjekteja käytettäessä ainoa viestin laukaisumenetelmä on aktiivinen kysely hallintalaitteen toimesta. Prosessisignaalit (*PDO, Process Data Object*) ovat CANopen-protokollan tarjoama tehokas tapa lähettää prosessitietoa väylälle. CAN-teknologia mahdollistaa prosessisignaalien käytön, koska mikä tahansa väylän laitteista voi lähettää tietoa väylälle. [3.] Prosessisignaalit mahdollistavat usean objektikirjaston alkion asettamisen osaksi samaa CAN-viestiä, jossa on maksimissaan kahdeksan tavua tietosisältöä.

Prosessisignaalikommunikoinnissa on kaksi roolia: prosessisignaalin vastaanottoja ja lähettäjä. Prosessisignaalin vastaanottajan näkökulmasta kyseessä on vastaanottoprosessisignaali (*RPDO, Receive Process Data Object*) ja lähettäjän näkökulmasta lähetysprosessisignaali (*TPDO, Transmit Process Data Object*). [6, s. 78.] Prosessisignaalien tuottaja-kuluttuja -periaate on esitetty kuvassa 3.7.



Kuva 3.7. *Prosessisignaalien periaate.*

Kuten kuvasta huomataan, tietyllä prosessisignaalilla voi olla yksi tuottaja, mutta monta kuluttajaa. Prosessisignaalien tuottaja-kuluttaja -periaatteen avulla saadaan ratkaistua palveluobjektien aiheuttamat ongelmat prosessitiedonsiirrossa; kuvassa laitteet B, C ja D saavat prosessitiedon suoraan laitteelta A ilman, että tiedon täytyy kiertää hallintalaitteen kautta.

Lähetys- ja vastaanottoprosessisignaaleille määritellään kommunikointiparametrit objektikirjastossa. Kommunikointiparametrit kuvaavat prosessisignaalien lähetykseen ja vastaanottoon liittyviä yksityiskohtia. Ne eivät ota kantaa, mitä tietoa prosessisignaali sisältää. Kommunikointiparametrit määrittävät esimerkiksi, millä viestitunnisteella tietty prosessisignaali lähetetään tai vastaanotetaan, mitä laukaisumenetelmää prosessisignaalin kanssa käytetään ja kuinka usein prosessisignaali pitäisi lähettää tai vastaanottaa [4, s. 77–79].

Prosessisignaalit voivat sisältää tietoa useista objektikirjaston alkioista. Prosessisignaalien kuvauksien (*PDO Mapping*) avulla määritellään, mitä objektikirjaston alkioita tietty prosessisignaaliviesti sisältää. Kuvauksen avulla prosessisignaalin tuottaja osaa muodostaa viestin oikein ja kuluttuja tietää, mitä prosessisignaali sisältää. Prosessisignaaliin voidaan kuvata mitä tahansa objektikirjaston alkioita. Ainoa rajoite on, että prosessisignaalin sisällä voi olla korkeintaan kahdeksan tavua tietoa. [8, s. 5–7.] Prosessisignaalien tietosisältö voi olla myös vähemmän kuin kahdeksan tavua eli kaikkia CAN-kehyksen tietosisältökenttiä ei tarvitse käyttää.

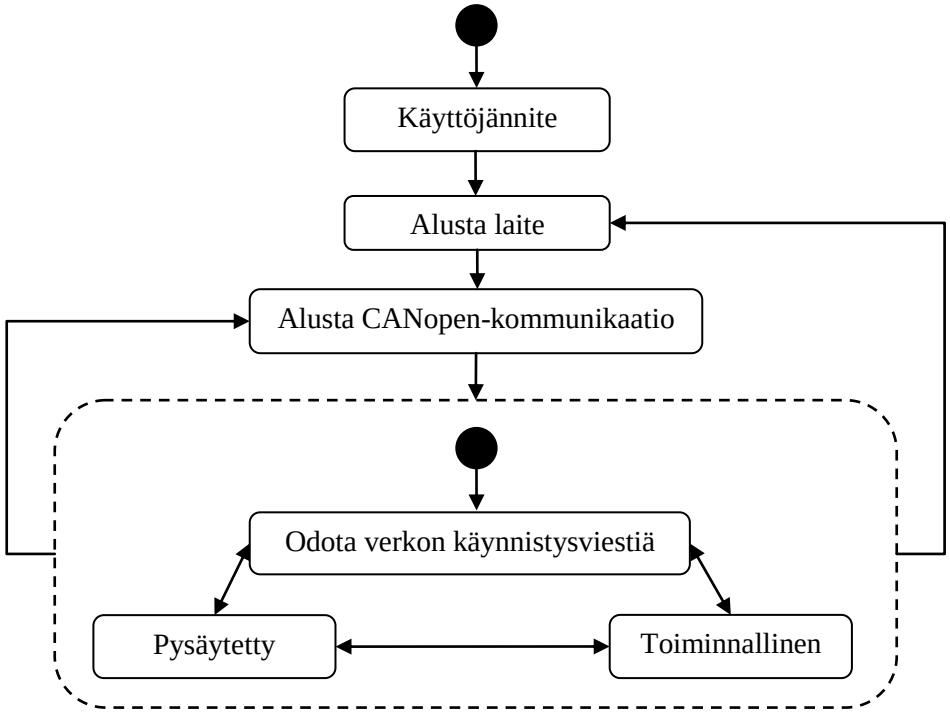
CANopen-protokolla tarjoaa useita mahdollisuuksia laukaista prosessisignaali väylälle. Tämä on selkeä etu verrattuna palveluobjektien käyttöön. CANopen tukee kokonaisuudessaan neljää eri laukaisumenetelmää [3]:

1. Tapahtumapohjainen
2. Aikapohjainen
3. Aktiivinen kysely
4. Tahdistettu

Usein laukaisumenetelmänä käytetään jotakin kombinaatiota näistä neljästä menetelmästä. Tapahtumapohjaisessa laukaisussa prosessisignaali lähetetään väylälle, mikäli jonkin prosessisignaaliin sijoitetuista objektikirjaston alkioista arvo muuttuu. Tapahtumapohjainen laukaisu on hyvin epädeterministinen, koska usein on mahdotonta ennustaa, koska prosessisignaali laukeaa. [4, s. 68–69.] Aikapohjaisessa laukaisussa prosessisignaalia lähetetään väylälle määritellyin väliajoin. Aikapohjainen menetelmä on hyvin deterministinen eli sitä käytettäessä voidaan tarkasti ennustaa väyläkuorma, suorituskyky sekä viiveet. [6, s. 79.] Prosessisignaali voidaan laukaista myös aktiivisen kyselyn avulla eli se laukaistaan, kun sitä pyydetään laitteelta. Suurin haitta kuitenkin on, että useat laitevalmistajat eivät tue kyseistä menetelmää tai toteuttavat sen eri tavalla. Niinpä kyseisen menetelmän käyttöä tulisi välttää. [4, s. 70.] Tahdistettua laukaisua käytetään, kun toimintojen tarvitsee tapahtua mahdollisimman samanaikaisesti. Usein esimerkiksi robotin ohjauksessa tarvitaan samanaikaisia toimintoja. Tahdistuminen toteutetaan tahdistusviestien avulla, jotka esitellään tarkemmin luvussa 3.6.3. Pääperiaate on hyvin yksinkertainen: kun tahdistusviesti vastaanotetaan, lähetetään prosessisignaali väylälle. [6, s. 79.]

3.6 Tietoverkon hallinta

CANopen-protokolla sallii tietoverkon hallintaisännän (*NMT Master, Network Management Master)* tarkkailla kaikkia tietoverkon laitteita ja sitä, että ne toimivat niille asetettujen parametrien mukaan [4, s. 83–86]. Hallintalaite voi käskeä jokaista laitetta erikseen tai kaikkia kerralla verkonhallintaviestien avulla ja näin määrätä, missä tilassa orjalaite on. Kuvan 3.8 tilakaavio esittää orjalaitteen mahdolliset tilat.



Kuva 3.8. *Orjalaitteen tilat [5, s. 8].*

Kuvassa lähdetään liikkeelle siitä, että CANopen-laitteelle ja -verkolle kytketään käyttöjännite. Tämän jälkeen jokainen laite hoitaa itsenäisesti alustustoimenpiteet. CANopen -kommunikaatio alustetaan muiden alustustoimien jälkeen. Kuvassa mustalla katkoviivalla merkitty alitilakone edustaa laitteen pysyviä tiloja. Alustustilat ovat hetkellisiä tiloja ja niiden suorittamisen jälkeen laite siirtyy automaattisesti odottamaan verkon käynnistysviestiä. Tilasiirtymässä CANopen-kommunikaation alustuksesta verkon käynnistysviestin odotukseen, lähettää laite väylälle viestin, joka ilmaisee, että laite on käynnistynyt onnistuneesti [5, s. 8]. Verkon käynnistysviestiä odotettaessa, CANopen-verkko on turvallisessa tilassa. Tämän tilan aikana CANopen-hallintalaite suorittaa väylän hallitun alustamisen, erinäisiä tarkistuksia ja mahdollisia asetusmuutoksia. Kun hallintalaite on suorittanut tarvittavat toimenpiteet ja tarkistukset, se siirtää verkon laitteet toiminnalliseen tilaan. Toiminnallinen tila on laitteiden ja verkon normaalitoimintaa vastaava tila. Pysäytettyyn tilaan laite voidaan ohjata vakavan virheen seurauksena.

Kuten kuvasta 3.8 huomataan, voidaan mistä tahansa pysyvästä tilasta siirtyä kumpaan tahansa alustustilaan, minkä jälkeen siirrytään automaattisesti odottamaan verkon käynnistysviestiä. Siirtymät kaikkien pysyvien tilojen välillä ovat myös mahdol-

lisia. Taulukossa 3.4 on esitetty, mitkä toiminnallisuudet ovat aktiivisia laitteen pysyvissä tiloissa.

Taulukko 3.4. *CANopen-toiminnallisuudet orjalaitteen pysyvissä tiloissa [5, s. 9].*

Viestityyppi	**Odota verkon käynnistystä**	**Toiminnallinen**	**Pysäytetty**
Palveluobjektit	x	x	
Hätäviestit	x	x	
Tahdistus	x	x	
Heartbeat	x	x	x
Laitteen vartiointi	x	x	x
Hallintaviesti	x	x	x
Prosessisignaalit		x	

Taulukosta huomataan, että prosessisignaalit ovat sallittuja vain toiminnallisessa tilassa. Muissa pysyvissä tiloissa toiminnallisuutta on rajoitettu erityisesti turvallisuussyistä.

3.6.1 CANopen-laitteen tilan valvonta

CANopen-hallintalaitteella on kaksi mahdollisuutta tarkkailla orjalaitteen tilaa. Laitteen vartioinnissa (*Node Guarding*) hallintalaite lähettää jokaiselle orjalaitteelle erikseen palveluobjektin avulla tilatietopyynnön, johon orjalaite vastaa omalla tilatiedollaan [5, s. 8]. Kyseisessä menetelmässä on kuitenkin kaksi huonoa puolta. Ensinnäkin kaistanleveyttä käytetään turhaan, koska yhden tilatiedon saamiseksi väylällä liikkuu kaksi viestiä: pyyntö ja vastaus. Tämän lisäksi ainoastaan hallintalaite voi kuluttaa tilatietovastauksen. Näistä syistä laitteen vartiointia ei nykyaikaisissa CANopen-verkoissa enää käytetä.

Laitteen vartiointia tehokkaampi ratkaisu on käyttää heartbeat-menetelmää. Menetelmän perusidea on, että jokainen laite lähettää tietyin väliajoin väylälle oman tilatietonsa ilman, että kenenkään tarvitsee sitä erikseen pyytää [5, s. 8]. Näin kaikki tilatietoa tarvitsevat laitteet voivat kuluttaa tiedon väylältä ja kaistanleveyttä ei mene hukkaan.

3.6.2 Hätäviestit

Hätäviestit mahdollistavat laitteen sisäisten virhetilojen ilmaisemisen. Kun verkon muut laitteet vastaanottavat hätäviestin, ne arvioivat sen sisällön ja tekevät tarvittavat toimenpiteet, jotka ovat valmistajariippuvaisia. Hätäviesti lähetetään vain kerran. [6, s. 84.]

Hätäviesti on yhden CAN-viestikehyksen mittainen ja oletuksena hätäviestin tunniste on laitteen laitetunnisteen ja luvun 0x80 summa [6, s. 138]. Koska hätäviesti on yhden CAN-viestikehyksen mittainen, sen sisältönä voi olla korkeintaan kahdeksan tavua tietoa. Ensimmäinen tavu on laitteen virherekisterin eli objektikirjaston indeksin 0x1001 sisältö. Tavut kaksi ja kolme sisältävät hätävirhekoodin ja loput viisi tavua on varattu laitteen valmistajan määrittelemää käyttöä varten.

3.6.3 Tahdistusviesti

Tahdistusviestin avulla voidaan aikaan saada verkon laitteiden samanaikainen käyttäytyminen. Tahdistusviesti lähetetään tietyin väliajoin ja se kertoo tahdistusviestin vastaanottajalle, että se voi aloittaa toiminnot, jotka on sidottu kyseiseen tahdistusviestiin. [3.] Tahdistusviesti toimii tuottaja-kuluttaja- periaatteella siten, että ainoastaan yksi laite voi tuottaa tietyn tahdistusviestin, mutta useat laitteet voivat kuluttaa sen.

Tahdistetut prosessisignaalit käyttävät tahdistusviestiä laukaisumenetelmänä. Tahdistetut prosessisignaalit täytyy lähettää tahdistusaikaikkunan sisällä. Tahdistusaikaikkuna on tietty ajanjakso siitä hetkestä, kun tahdistusviesti on vastaanotettu. [4, s. 70–73.] Tahdistusviestiä voidaan käyttää myös vastaanotetun prosessisignaalin arvon käyttöönottoon. Tällöin uusi prosessisignaalin arvo tulee ottaa käyttöön tahdistusaikaikkunan sisällä.

4 CAN-POHJAISET YLEMMÄN TASON PROTOKOLLAT

CAN-protokolla antaa pohjan monelle etenkin teollisuudessa paljon käytetylle ylemmän tason protokollalle. Ylemmän tason protokollat tarjoavat joustavuutta, toimintoja ja vaihtoehtoja järjestelmän rakentamiseen ja konfigurointiin verrattuna pelkän CAN-protokollan käyttöön. Perinteisesti CAN-verkoissa käytetään tiettyä ylemmän tason protokollaa ja ne on tarkoitettu sulautettuihin ja suljettuihin sovelluksiin. Nykyisin kuitenkin on usein tarve integroida CAN-verkkoja osaksi suurempia järjestelmiä, mikä on mahdollista protokollamuuntimina toimivien yhdyskäytävien avulla. Esimerkiksi CiA-järjestö määrittelee kaksi standardia, jotka määrittelevät CANopen-yhdyskäytävän SAE J1939 -pohjaisiin sekä Ethernet-pohjaisiin verkkoihin. CiA DSP-309 standardi määrittelee CANopen-yhdyskäytävän TCP/IP- ja Ethernet-pohjaisiin verkkoihin [9, s. 5]. CiA DSP-413 mahdollistaa CANopen-yhdyskäytävän, jonka avulla voidaan liittää CANopen-laitteita osaksi SAE J1939 -pohjaisia tietoverkkoja [10, s. 4]. Standardien avulla toteutetut yhdyskäytävät suorittavat muunnoksen protokollasta toiseen. Näiden standardien avulla mahdollistetaan hierarkkisten verkkoratkaisuiden toteutus ja suunnittelu ilman, että kaikissa verkon segmenteissä tulee olla sama protokolla. Eri protokollien välisiä muuntimia ja yhdyskäytäviä on saatavana myös kaupallisina laitteina.

Koska usein CAN-tietoverkkojen yhteydessä käytetään muitakin ylemmän tason CAN-protokollia kuin CANopen, on hyvä perehtyä kyseisten protokollien perusominaisuuksiin. Tämän työn kannalta erityisen mielenkiintoista on, mitä yhtäläisyyksiä ja eroja muilla CAN-pohjaisilla ylemmän tason protokollilla on verrattuna CANopen-protokollaan. Näihin kiinnitetään erityistä huomiota seuraavissa luvuissa, jotka esittelevät neljä laajasti käytettyä protokollaa: DeviceNET, CAN Kingdom, Smart Distributed System (SDS) sekä SAE J1939.

4.1 DeviceNET

DeviceNET on seitsemän kerroksiseen OSI-referenssimalliin perustuva ylemmän tason protokolla, joka on tarkoitettu erityisesti tietoverkoksi teollisuuden ohjainten ja I/O-laitteiden (*Input/Output*) välille. DeviceNET-protokollan sovellusalueet painottuvat vahvasti tehdasautomaatioon. DeviceNET on yhdistelmä ylemmän tason CIP-protokollasta (*Common Industrial Protocol*) ja CAN-protokollan fyysisestä tasosta [11, s. 1]. CIP-protokollassa tietoverkon laitteet esittävät itsensä joukkona objekteja CANopen-protokollan tapaan [12]. Myös CIP-protokollassa määritellään objekteja, jotka jokaisen laitteen on pakko toteuttaa. Näihin lukeutuvat muun muassa objektit laitteen tunnistamiseen sekä laitteen asetuksiin kuten baudinopeuteen ja laitetunnisteeseen. CANopen-protokollan tapaan myös DeviceNET määrittelee pakollisten objektien lisäksi useita laiteprofiileja sekä elektronisen tietolehden.

DeviceNET salli verkossa 64 laitetta ja verkossa on kaksi pääroolia: isäntä ja orja. Isäntälaite voi omistaa kaikki tai ainoastaan sen tarvitsemat orjat. Orjalaitteen voi omistaa ainoastaan yksi isäntälaite kerrallaan ja orjalaite myöntää omistuksen tietylle isäntälaitteelle, kun isänlaite sitä pyytää [11, s.7]. Orjalaite voi myös kieltää omistuksen, mikäli se kuuluu jo toisella isäntälaitteelle. Verkossa sallitaan vakiona useita isäntälaitteita toisin kuin CANopen-protokollassa.

DeviceNET vaatii kommunikaatiokanavan laitteiden välille, mikäli ne haluavat vaihtaa tietoa toisin kuin CANopen-protokolla. DeviceNET tukee kolmea kommunikointityyppiä: päästä päähän kommunikointi, eksplisiittinen kommunikointi sekä I/O-kommunikointi. Päästä päähän kommunikointi voi tapahtua minkä tahansa kahden DeviceNET-laitteen välillä. Tiedon vaihdossa ei oteta kantaa, mitä viestien tietosisältö tarkoittaa tai miten sitä tulee tulkita; kommunikoivien laitteiden tulee yksinkertaisesti tietää, mitä viestien sisältö tarkoittaa [12]. Tämän takia päästä päähän kommunikointia käytetään yleensä ainoastaan saman valmistajan laitteiden välisessä kommunikaatiossa. Eksplisiittinen tiedon vaihto tapahtuu aina isäntä- ja orjalaitteen välillä. Sen pääasiallinen tarkoitus on tarjota isäntälaitteelle mahdollisuus orjalaitteen konfigurointiin, mutta niiden avulla voidaan lukea myös laitteelta prosessitietoa [13]. Niiden voidaan siis ajatella vastaavan CANopen-protokolla palveluobjekteja. I/O-kommunikointi tapahtuu myös isäntä- ja orjalaitteen välillä. Sen avulla vaihdetaan prosessitietoa, joten ne vastaavat CANopen-protokollon prosessisignaaleita. DeviceNET mahdollistaa I/O-viesteissä paloitellun tiedon siirron eli sen avulla voidaan siirtää prosessitietoa suurempia määriä kuin kahdeksan tavua [13]. CANopen-mahdollistaa prosessitiedon siirrossa maksimissaan kahdeksan tavua.

DeviceNET tarjoaa palvelun, joka takaa, että verkossa on tietyllä laitetunnisteella ainoastaan yksi laite. Näin varmistetaan, että tietoverkon toiminta ei häiriinny. Kun tietoverkko käynnistyy, jokainen laite suorittaa kyselyn, jossa se tarkistaa, onko verkossa jo samalla laitetunnisteella olevaa laitetta. [14.] Mikäli laitetunniste on jo käytössä, laitetunnistekyselyn tehnyt laite ei saa lupaa kytkeytyä väylään. CANopen-protokolla ei tarjoa kyseistä toiminnallisuutta.

Lian et al. [15] esittävät tutkimuksessaan vertailun kolmen tietoverkkotekniikan välillä: Ethernet, ControlNET ja DeviceNET. Vertailuja suoritetaan erityisesti ohjaustietoverkkojen ominaisuuksien ja vaatimusten pohjalta. Tutkimuksen mukaan DeviceNET sopii erityisesti ohjausverkkoihin, joissa käytetään priorisoituja ja lyhyitä viestejä. Tutkimus myös osoittaa, että CAN-teknologiaan pohjautuvat protokollat ovat hyvin deterministiä eli niiden toimintoja ja viiveitä voidaan hyvin ennustaa.

4.2 CAN Kingdom

CAN Kingdom on kehitetty erityisesti hajautettujen ja sulautettujen ohjausjärjestelmien tarpeita varten. Suuri ero CAN Kingdomin ja CANopen-protokollan välillä on, että CAN Kingdom ei perustu OSI-referenssimalliin. OSI-mallin periaate on, että kehittäjän ei tarvitse tietää kerroksista mitään muuta, kuin rajapinnat ylempään ja alempaan kerrokseen. Tämä ei kuitenkaan pidä paikkaansa reaaliaikaisuutta vaativissa, sulautetuissa verkoissa, koska niitä kehitettäessä tulee ymmärtää kommunikaatio kaikkien kerroksien läpi [16, s. 4]. CAN Kingdom-protokollan kehittäjien mielestä OSI-malli ei sovellu ohjaustietoverkkoihin, koska se on tarkoitettu kommunikaatiotietoverkkoihin, joissa käyttäjien määrää ei voida tietää tietoverkon suunnittelun aikana ja reaaliaikavaatimukset eivät ole niin kovia kuin ohjaustietoverkoissa [16, s. 1].

Kun laite yhdistetään CAN Kingdom-verkkoon, sillä ei ole oikeutta tehdä mitään, ennen kuin se saa luvan tietoverkon kuninkaalta (*The King*) [14]. Kuninkaita voi olla ainoastaan yksi, joten se voidaan rinnastaa CANopen-protokollan hallintalaitteeseen, mutta kuningas toimii osana verkkoa ainoastaan verkon alustuksen aikana. Kun alustus on suoritettu, kuningas ei puutu tietoverkon kommunikaatioon [14]. Kuninkaan vastuulla on organisoida koko tietoverkon kommunikaatio ja se on ainoa laite, joka tietää miten tietoverkon tulisi toimia. Tietoverkon käynnistyessä kuningas jakaa viestitunnisteet kaikille verkon laitteille, joten ne eivät oleta omistavansa mitään CAN-viestitunnisteita, toisin kuin esimerkiksi CANopen-protokollassa [14]. Näin tietoverkon laite tietää, mitä viestitunnisteita sen tulee kuunnella ja mitä viestitunnisteita sen tulee käyttää tiedon lähetykseen. Tietoverkon kuningas mahdollistaa tietoverkon suunnittelijalle täyden kontrollin kommunikaatiosta. Kun kuningas on suorittanut verkon alustamisen, alkaa väylän normaalitoiminta.

CAN Kingdom tarjoaa DeviceNET-protokollan tapaan mahdollisuuden tarkistaa, onko verkossa liitettynä laitteita samalla laitetunnisteella. Kuningas laite tekee tarkistuksen verkon käynnistyessä ja mikäli se havaitsee laitteita samoilla laitetunnisteilla, kuningas voi asettaa laitteille eri laitetunnisteet tai vaihtoehtoisesti poistaa laitteita käytössä [14]. Kuningas voi myös estää koko verkon käynnistyksen, mikäli kuninkaalle tuntemattomia laitteita on liitettynä verkkoon tai verkosta puuttuu laitteita.

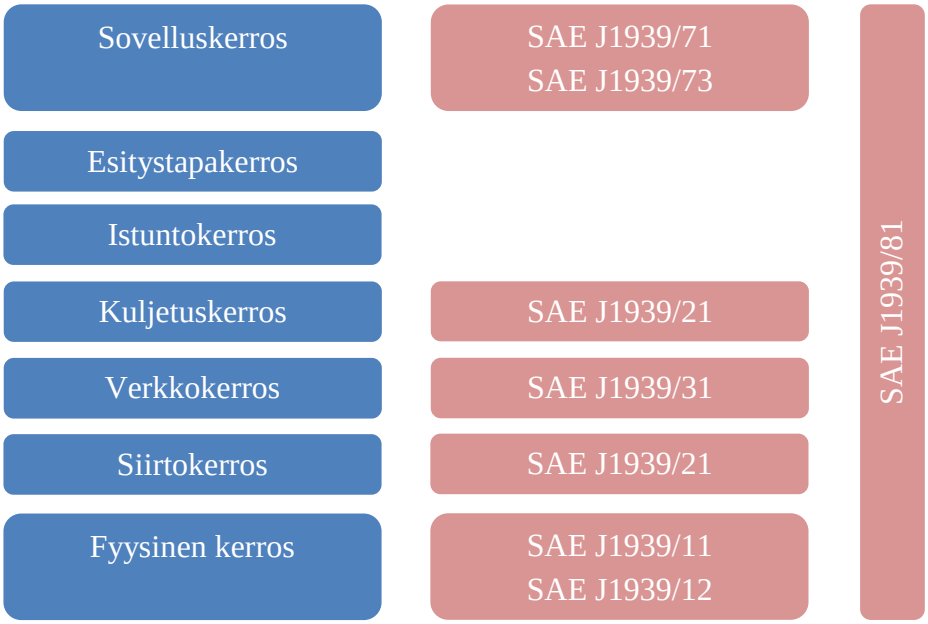
4.3 Smart Distributed System (SDS)

Smart Distributed System (SDS) -protokollan avulla voidaan liittää I/O-laitteita ohjelmoitaviin logiikoihin. SDS-protokolla on alun perin suunniteltu käytettäväksi hajautettujen binäärisensoreiden ja -toimilaitteiden kanssa. SDS-protokollan kommunikointi tapahtuu pääasiallisesti isäntä- ja orjalaitteen välillä ja isäntälaitteella on koko ajan 100 % kontrolli koko verkosta [14]. Isäntälaite suorittaa verkon käynnistyessä tarkistuksia, joissa se tutkii, ovatko kaikki verkon laitteet paikalla ja toiminnassa. SDS pohjautuu OSI-referenssimalliin ja se on sovelluskerroksen protokolla, joka rakentuu suoraan siirtokerroksen päälle. SDS-protokolla ei ole sidottu käyttämään CAN-standardin siirto- ja fyysistä kerrosta, vaan sitä voidaan helposti käyttää myös esimerkiksi Ethernet-pohjaisten verkkojen kanssa [17, s. 12].

Prosessidatan lähetykseen SDS-protokolla määrittelee kaksi viestityyppiä: lyhyen ja pitkän viestin. Lyhyessä viestissä ei ole ollenkaan tietosisältöä eli tietosisällön pituuskentän tulee aina olla nolla. Tietosisällön sijaan tieto sisältyy käytettyyn 3-bittiseen palvelutyyppiin, joka on osa CAN-viestitunnistetta. [13.] Lyhyt viesti on tarkoitettu käytettäväksi binäärisen tiedon kanssa. Mikäli kyseessä ei ole binäärinen tieto, täytyy käyttää pitkää viestityyppiä, jossa käytetään hyväksi CAN-viestin tietosisältöä. Pitkää viestiä käytettäessä tulee aina tietosisällön pituus kentän olla suurempi kuin nolla. [17, s. 44.] Näin voidaan erottaa lyhyt ja pitkä viesti toisistaan. SDS tarjoaa CANopen-protokollan tapaan mahdollisuuden konfiguroida laitteita kahden laitteen välisten viestien avulla. Ennen kuin konfigurointi voidaan suorittaa, pitää kuitenkin muodostaa dynaamisesti yhteys laitteiden välille SDS-protokollan yhteyksien hallintalaitteen avulla (*Connection Manager*). Laite voi pyytää yhteyksien hallintalaitteelta luvan suorittaa operaatioita tietylle laitteelle. [13.] Vasta luvan saatuaan laite voi suorittaa haluamansa operaatiot.

4.4 SAE J1939

SAE J1939-protokolla on SAE-järjestön (*Society of Automotive Engineers*) määrittelemä protokolla, joka on alun perin suunniteltu erityisesti ajoneuvoteollisuuden tarpeisiin, joten SAE J1939-protokollan pääasialliset käyttötarkoitukset eroavat CANopen-protokollasta. SAE J1939-protokolla ei määrittele laiteprofiileita tai elektronisia tietolehtiä. SAE J1939-standardi perustuu OSI-referenssimalliin kuten CANopen-protokollakin. Jakautuminen eri OSI-referenssimallin tasoille on esitetty kuvassa 4.1.



Kuva 4.1. *SAE J1939-standardin sijoittuminen OSI-referenssimalliin [18].*

SAE J1939-standardi koostuu useista osista, jotka määrittelevät eri kerrosten ominaisuuksia. Standardi ei kuitenkaan määrittele esitystapakerroksen tai istuntokerroksen toiminnallisuuksia. Fyysisellä kerroksella määritellään väylänmaksimipituudeksi 40 metriä, väylänopeudeksi 250 kilobaudia ja väylä topologiaksi lineaarinen väylä [19, s. 29]. Laitteita SAE J1939-pohjaisessa väylässä voi olla korkeintaan 30. SAE J1939/21-standardi, joka sijoittuu siirto- ja kuljetuskerroksille, määrittelee erityisesti SAE-standardin käyttämän viestiformaatin [20]. Verkkokerroksen standardi SAE J1939/31 määrittelee SAE J1939-standardin mukaisen protokollasillan, jonka avulla voidaan muodostaa lineaarista väylää monimutkaisempia väylätopologioita [18]. SAE J1939/81-standardi määrittelee laitteelle nimen sekä osoitteen.

SAE J1939-standardi käyttää CAN-viesteissä laajennettua 29-bittistä viestitunnistetta toisin kuin CANopen-protokolla, joka käyttää 11-bittistä viestitunnistetta. SAE J1939-protokollan mukaisen 29-bittisen viestitunnisteen sisältämä informaatio on esitetty kuvassa 4.2.

Prioriteetti (3 bittiä)	Parametrin ryhmänumero (18 bittiä)	Lähettäjän osoite (8 bittiä)

Kuva 4.2. *SAE J1939-standardin 29-bittinen CAN-viestitunniste [21, s. 3].*

Prioriteettikentällä määrätään viestien prioriteetti kolmella bitillä. Korkeimmat prioriteetit myönnetään viesteille, jotka kontrolloivat järjestelmän toimintaa. Parametrin ryhmänumero (*PNG, Parameter Number Group*) on 18 bitin arvo, joka sisältää tietoa esimerkiksi viestinlähetystyypistä ja mahdollisesta vastaanottajan osoitteesta [21, s. 3–4]. Lähes kaikki SAE J1939-protokollan viestit ovat lähetystyypiltään yleislähetyksiä, joiden sisällä lähetetään prosessitietoa. Tämä helpottaa järjestelmän suunnittelua ja viestitunnisteiden määrittelyä, koska näin ei tarvitse erikseen osoittaa viestiä tietylle laitteelle [20]. Viestit voidaan lähettää myös kahden laitteen välillä. Parametri ryhmän numeron avulla voidaan päätellä, minkä tyyppistä tietoa ja kuinka paljon viesti sisältää [20]. Viimeiset kahdeksan bittiä viestitunnisteessa sisältävät lähettävän laitteen osoitteen.

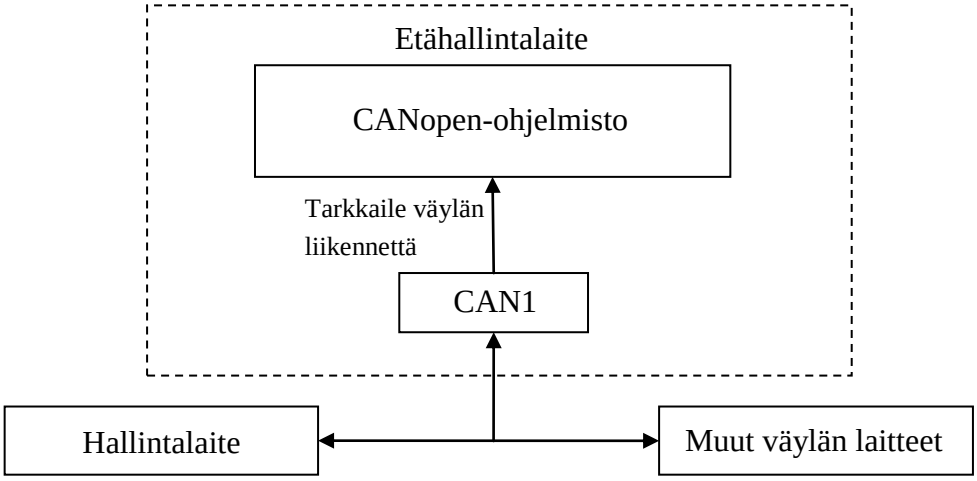
SAE J1939-protokolla määrittelee menetelmän, jonka avulla laitteen osoite voidaan määrittää dynaamisesti. CANopen-protokolla ei tue kyseistä ominaisuutta. Menetelmää kutsutaan osoitteen vaatimiseksi (*Address Claim*). Laite voi vaatia tiettyä osoitetta väylän laitteilta lähettämällä pyynnön väylälle. Mikäli jollakin laitteella on jo kyseinen osoite käytössä, lähetetään pyynnön esittäneelle laitteelle tästä tieto, joka kertoo, että laite ei voi ottaa osoitetta käyttöön. Laite voi myös ensin kysyä väylältä kaikkien siihen liitettyjen laitteiden osoitteet. [20.] Väylän laitteet vastaavat kyselyyn ja näin laite voi tutkia, mikä on ensimmäinen vapaa osoite väylällä. Tämän jälkeen laite pyytää ensimmäistä vapaata osoitetta itselleen.

SAE J1939-standardit toimivat pohjana ISOBUS-protokollalle sekä NMEA 2000-protokollalle (*National Marine Electronics Association*). ISOBUS-standardi koostuu useista osista ja se määrittelee tiedonsiirron, jonka avulla voidaan kontrolloida ja kommunikoida erityisesti traktoreiden ja siihen liitettävien laitteiden kanssa [22]. NMEA 2000-protokolla on tarkoitettu käytettäväksi vesille liikkuen laitteiden ja koneiden yhteydessä [23].

5 MAHDOLLISUUDET LIITTÄÄ ETÄHALLINTALAITE OSAKSI CANOPEN-JÄRJESTELMÄÄ

Etähallintalaite voidaan liittää osaksi CANopen-järjestelmää usealla eri tavalla. Tapa, jota käytetään, riippuu siitä, mitä vaatimuksia etähallintalaitteen toiminnallisuudella asetetaan. Tässä työssä vaatimuksena ovat prosessisignaalien ja hätäviestien kuuntelu sekä palveluobjektioperaatioiden suoritus. Prosessisignaaleita halutaan kuunnella, koska niiden arvot voidaan lähettää etähallintapalvelimelle, josta niitä voidaan tarkkailla ja käsitellä erilaisten algoritmien avulla. Hätäviestit ovat kriittisiä järjestelmän toiminnan kannalta, joten niistä halutaan tieto palvelimelle, jotta voidaan esimerkiksi ennalta varautua huoltotoimenpiteisiin. Kun mahdollistetaan palveluobjektioperaatiot etähallintalaitteen avulla, mahdollistetaan järjestelmän konfigurointi helposti ilman, että esimerkiksi huoltomiehin täytyy lähteä paikan päälle järjestelmän asetuksia muuttamaan.

Passiivinen kuuntelu on yksinkertaisin mahdollinen tapa liittää laite osaksi CANopen-järjestelmää. Periaate on esitetty kuvassa 5.1.



Kuva 5.1. *Passiivinen kuuntelu väylän liikenteestä.*

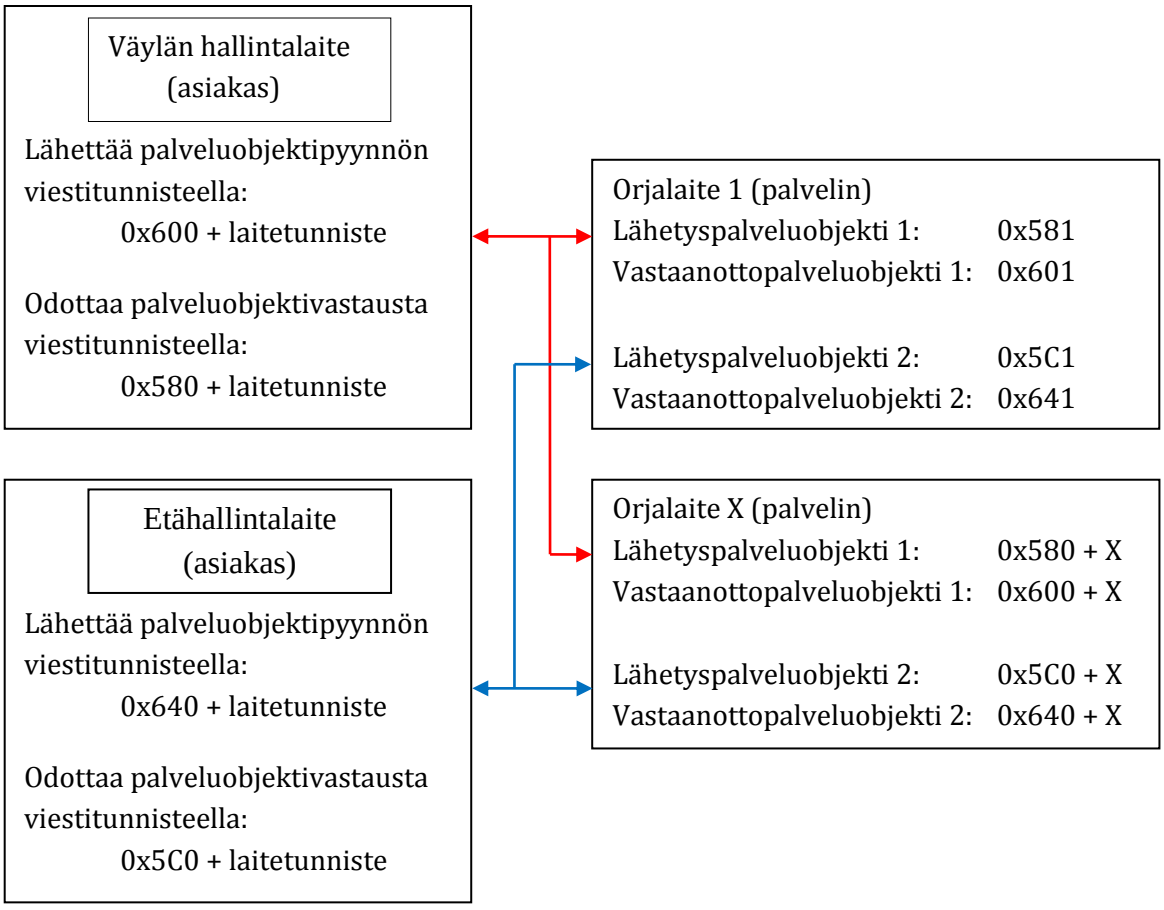
Passiivinen kuuntelu mahdollistaa väylällä liikkuvien viestien kuuntelun. Tällöin etähallintalaite ei kuitenkaan voi tehdä palveluobjektioperaatioita, koska oletuksena väylän alkuperäinen hallintalaite omistaa kaikki palveluobjektikanavat. Väylän toiminta saattaisi häiriintyä, mikäli väylän alkuperäinen hallintalaite näkisi etähallintalaitteen tekemät palveluobjektioperaatiot, koska CANopen-standardi ei määrittele, miten alkuperäisen hallintalaitteen tulisi tässä tilanteessa toimia. On myös vaikea ennustaa, miten orja-

laite toimii, mikäli useampia laitteita suorittaa sille palveluobjektioperaatioita samanaikaisesti samalla viestitunnisteella. Kuvassa esitetty passiivinen kuuntelu on kuitenkin tärkeä osa etähallintalaitetta; prosessisignaalien ja hätäviestin kuuntelu ja talletus voidaan ratkaista passiivisen kuuntelun avulla.

Seuraavat aliluvut esittelevät mahdollisuuksia etähallintalaitteen liittämiseksi osaksi CANopen-verkkoa siten, että mahdollistetaan sekä viestien kuuntelu että palveluobjektioperaatiot. Oletuksena on, että väylässä on valmiiksi kiinni laite, joka toimii CANopen-hallintalaitteena. Useat menetelmistä perustuvat siihen, että väylän alkuperäinen hallintalaite edelleen huolehtii kriittisistä toiminnoista, kuten verkon käynnistyksestä ja hätäviesteihin reagoinnista. Tärkeimpänä kriteerinä menetelmän valitsemiselle voidaan pitää sitä, että menetelmä ei vaadi muutoksia alkuperäiseen väylään. Tämä on erityisen tärkeää ratkaisun geneerisyyden kannalta, koska etähallintaratkaisun tavoitteena on, että samaa ratkaisua voidaan käyttää kaikissa CANopen-järjestelmissä eli tiettyyn järjestelmään sidottuja ratkaisuja pyritään välttämään.

5.1 Palveluobjektikanavien lisääminen

Orjalaite voi sallia useita palveluobjektikanavia. Tällöin orjalaite toteuttaa useamman palveluobjektipalvelimen eli CAN-viestitunnisteen, jolla siihen voidaan suorittaa palveluobjektioperaatioita. [4, s. 63.] Kuva 5.2 havainnollistaa tilanteen.



Kuva 5.2. *Palveluobjektikanavien lisääminen.*

Kuvasta huomataan, että laitteet suorittavat palveluobjektioperaatiot eri viestitunnisteilla, jolloin konflikteja ei synny. Kuvassa palveluobjekteille varattu viestitunnisteavaruus on jaettu puoliksi väylän hallintalaitteen ja etähallintalaitteen kesken ja orjalaitteet toteuttavat kaksi palveluobjektipalvelinta. Palvelimia voisi olla enemmänkin, jolloin palveluobjekteille varattu viestitunnisteavaruus jaettaisiin useampiin osiin. Tämä lisäisi laitteiden määrää, jotka saavat palveluobjektioperaatioita suorittaa.

Mikäli järjestelmän orjalaitteet valmiiksi sallivat useita palveluobjektikanavia, voidaan etähallintalaite lisätä helposti järjestelmän osaksi. Ongelmana on kuitenkin se, että hyvin harvat järjestelmät tukevat kyseistä ominaisuutta. Näin ollen lähes aina jokaiseen orjalaitteeseen ja alkuperäiseen hallintalaitteeseen täytyisi tehdä muutoksia. Palveluobjekteille varatun viestitunnisteavaruuden jakaminen myös rajoittaa väylän laitteiden lukumäärää. CANopen-protokolla sallii väylässä 127 laitetta. Sallittujen laitteiden määrä puolittuu aina palveluobjektipalvelimien määrän kaksinkertaistuessa; kun palveluobjekteille varattu viestitunnisteavaruus jaetaan esimerkiksi kahdelle laitteelle, sallitaan enää 64 laitetta. Lisäksi CANopen-standardissa ei ole määritelty, miten viestitunnisteavaruus tulisi jakaa, joten on hyvin vaikea tehdä yleiskäyttöistä ratkaisua.

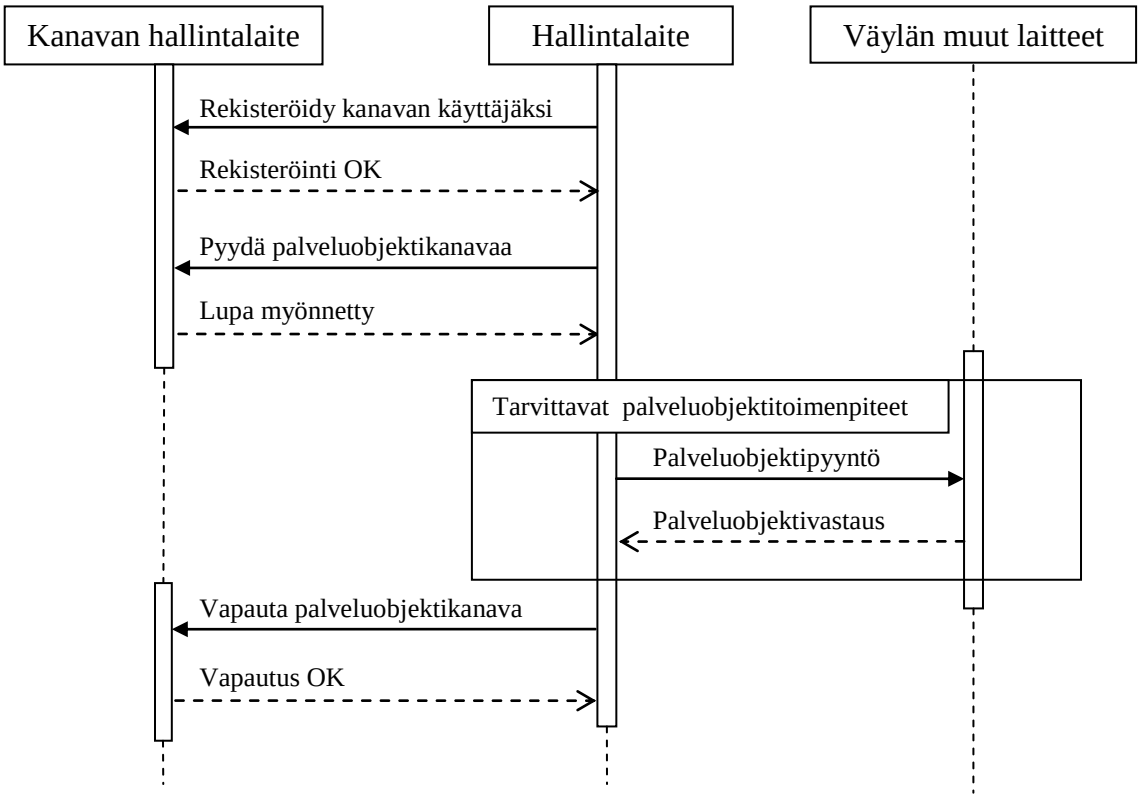
5.2 Nykyisen hallintalaitteen korvaaminen etähallintalaitteella

Nykyinen hallintalaite on mahdollista korvata kokonaan laitteella, joka on etähallittava. Tämä onkin varsin hyvä ratkaisu, koska näin vältytään mahdollisilta kahden hallintalaitteen tuomilta riskeiltä. Todennäköistä myös on, että sekä CANopen-verkon hankinta- että ylläpitokustannukset pienenevät laitteiden määrän vähentyessä yhdellä.

Huono puoli on kuitenkin se, että usein verkon hallintalaite toteuttaa myös muita toiminnallisuuksia kuin pelkästään CANopen-verkon hallinnan. Usein esimerkiksi koneen ohjausalgoritmeja ajetaan hallintalaitteella. Niinpä kyseiset toiminnallisuudet tulisi aina ohjelmoida etähallintalaitteeseen. Tämä kuitenkin johtaa siihen, että etähallintalaite olisi aina tiettyä tarkoitusta varten tehty, eikä sitä voisi suoraan siirtää osaksi jotakin muuta järjestelmää. Tämän työn tarkoituksena oli kehittää yleiskäyttöinen CANopen-verkon etähallintaratkaisu, joten tästä syystä nykyisen hallintalaitteen korvaaminen kokonaan etähallintalaitteella on suljettava pois. Mikäli oltaisiin rakentamassa täysin sovelluskohtaista CANopen-verkkoa, johon halutaan etähallinta, olisi tämä ratkaisu kuitenkin todennäköisesti paras vaihtoehto.

5.3 Jaettu palveluobjektikanava

Palveluobjektikanavat on mahdollista jakaa laitteiden kesken, mikäli CANopen-järjestelmän orjalaitteet tukevat ainoastaan yhtä palveluobjektikanavaa. Tällöin väylällä tarvitaan laite, joka pitää kirjaa palveluobjektikanavien käytöstä ja antaa palveluobjektikanavia tietyn laitteen käyttöön, kun laite niitä pyytää [4, s. 63]. Laitetta kutsutaan palveluobjektikanavan hallintalaitteeksi. Kuva 5.3 esittää jaetun palveluobjektikanavan periaatteen.



Kuva 5.3. *Jaetun palveluobjektikanavan periaate.*

Aluksi hallintalaite rekisteröityy palveluobjektikanavan käyttäjäksi. Kuvassa 5.3 hallintalaite edustaa sekä väylän alkuperäistä hallintalaitetta että etähallintalaitetta eli niiden molempien pitää rekisteröityä palveluobjektikanavan käyttäjiksi. Kun rekisteröinti on suoritettu onnistuneesti, voi hallintalaite pyytää käyttöönsä kaikkia palveluobjektikanavia tai ainoastaan niitä, joita se todella tarvitsee. Lupaa palveluobjektikanavan käyttämiseen ei välttämättä saa heti, jos jokin toinen laite on varannut saman palveluobjektikanavan käyttöönsä. Tällöin hallintalaitteen tulee pyytää lupaa uudellaan. Kun lupa on myönnetty, voi hallintalaite aloittaa palveluobjektitoimenpiteet. Onnistuneiden toimenpiteiden jälkeen hallintalaite vapauttaa varaamansa palveluobjektikanavat, jotta ne ovat vapaita muita rekisteröityjä palveluobjektikanavan käyttäjiä varten. [24.]

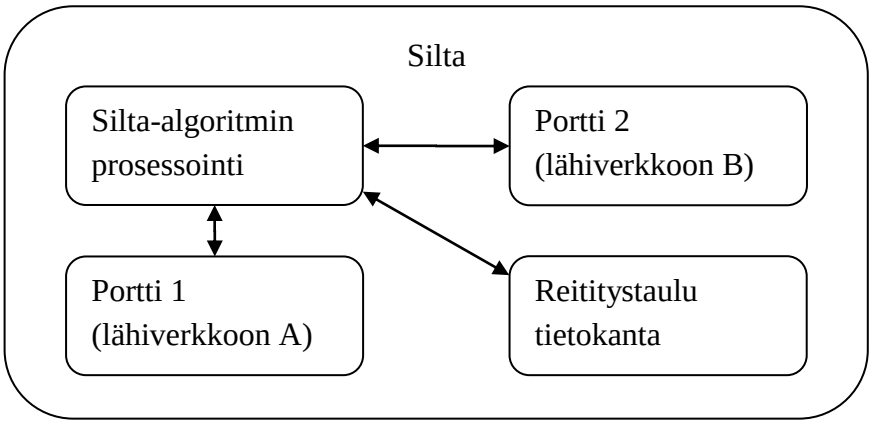
Jaettu palveluobjekti kanava on standardoitu tapa, jonka avulla useat laitteet voivat suorittaa palveluobjektioperaatioita. Jaettu palveluobjektikanava on määritelty CiA:n standardissa DSP-302-5 [24]. Standardi vaatii palveluobjektikanavan hallintalait-

teen väylälle. Tämä ei ole ongelma, koska palveluobjektikanavan hallinta voitaisiin toteuttaa osaksi etähallintalaitetta, jolloin kyseistä toiminnallisuutta ei tarvitsisi toteuttaa johonkin väylän alkuperäisistä laitteista tai väylään ei tarvitsisi lisätä erillistä laitetta, joka toteuttaa palveluobjektikanavan hallinnan. Näin ollen palveluobjektikanavan hallintalaitteen ja -logiikan lisääminen ei aiheuta muutoksia alkuperäiseen väylään.

Jaetun palveluobjektikanavan suurin ongelma on, että hyvin harvat väylän alkuperäiset hallintalaitteet tukevat jaetun palveluobjektikanavan käyttöä. Ne olettavat, että voivat käyttää palveluobjektikanavaa ilman, että niiden tarvitsee pyytää siihen erikseen lupaa. Ominaisuutta ei tueta, koska usein CANopen-järjestelmät ovat pieniä ja suljettuja järjestelmiä, joissa hallintalaitteita ja palveluobjektikanavia ei tarvita kuin yksi. Ominaisuuden lisääminen kasvattaa ohjelmakoodin kokoa ja monimutkaistaa hallintalaitteen toteutusta eli se lisää kustannuksia. Jaetun palveluobjektikanavan käyttö myös lisää palveluobjektioperaatioiden viivettä, koska palveluobjektikanavan käyttöön tulee pyytää erikseen lupa ennen varsinaisten operaatioiden suoritusta. Edellä mainituista syistä tuki jaetulle palveluobjektikanavalle täytyisi lähes aina toteuttaa etenkin jo olemassa oleviin, vanhoihin hallintalaitteisiin, mikä monissa tapauksissa on mahdotonta tai se ei kustannus syistä kannata.

5.4 CANopen-silta

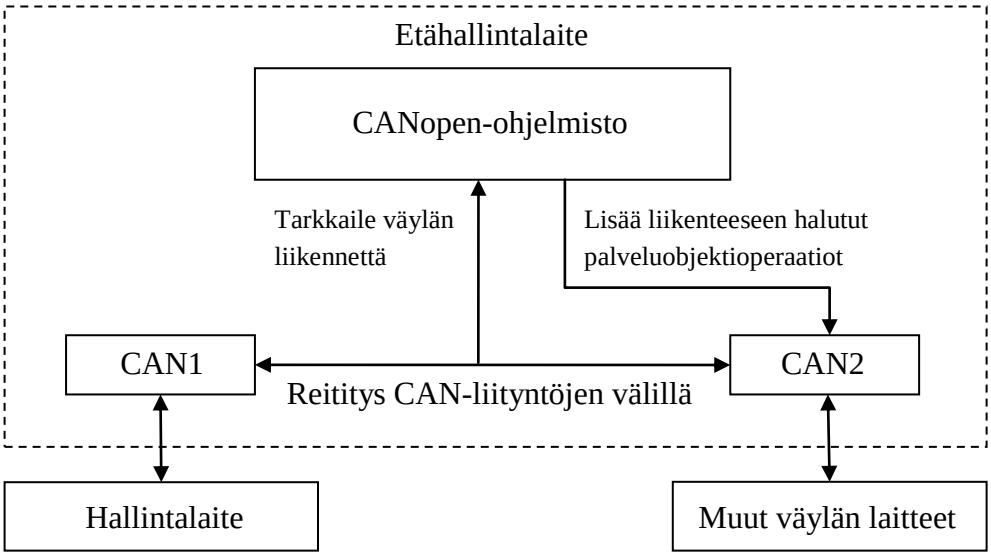
Protokollasiltoja käytetään paljon paikallisten lähiverkkojen arkkitehtuurin muodostamiseen. Sillat yhdistävät kaksi lähiverkkoa toisiinsa. Perustoiminnaltaan siltojen tarkoituksena on vastaanottaa viesti ja päätellä, pitääkö kyseinen viesti reitittää toiselle puolelle siltaa. Päättelyyn käytetään erityisiä reititystauluja, joihin on talletettuna tieto siitä, tarvitaanko tiettyä viestiä toisella puolella siltaa. [25, s. 508.] Periaate on esitetty alla olevassa kuvassa 5.4.



Kuva 5.4. *Lähiverkkojen yhteydessä käytetyn sillan periaate [25].*

Taulut muodostetaan siltojen käynnistyksen jälkeen, jolloin aluksi reitite tään kaikki viestit molemmille puolille ja algoritmien avulla päätellään, tarvittiinko kyseistä viestiä sillan molemmin puolin [25, s. 508]. Kyseinen tieto talletetaan reititystauluun ja tietoa käytetään jatkossa saman viestitunnisteen omaaville viesteille.

Siltaratkaisun periaatetta voidaan käyttää hyväksi myös CANopen-järjestelmissä. Siltaratkaisu vaatii etähallintalaitteelta kaksi erillistä CAN-liityntää, jotta reititys laitteen kautta voidaan toteuttaa. Siltaratkaisun periaate on esitetty kuvassa 5.5.



Kuva 5.5. *CANopen-silta toteutettuna kahden CAN-liitynnän avulla.*

Siltaratkaisussa etähallintalaitteen toiseen CAN-liityntään kytketään väylän alkuperäinen hallintalaite ja toiseen väylän orjalaitteet. Etähallintalaite reitittää viestit laitteiden välillä ja kuuntelee samalla väylän liikennettä. Se voi myös lisätä liikenteeseen omat palveluobjektioperaationsa. Etähallintalaitteen tekemiä palveluobjektioperaatioita tai niihin tulevia vastauksia ei kuitenkaan reititä väylän alkuperäiselle hallintalaitteelle. Näin varmistetaan, että väylän toiminta ei häiriinny etähallintalaitteen tekemien palveluobjektioperaatioiden seurauksena. Etähallintalaite voi suorittaa palveluobjektioperaatioita ainoastaan silloin, kun väylän varsinaisella hallintalaitteella ei ole palveluobjektioperaatioita käynnissä.

CANopen-siltaratkaisu ei ole sidottu mihinkään tiettyyn järjestelmään ja samaa ratkaisua voidaan etähallintalaitteen asetuksia muuttamalla käyttää kaikissa CANopen-järjestelmissä. Siltaratkaisu ei vaadi muutoksia alkuperäiseen väylään. Viestien reititys etähallintalaitteen kautta kuitenkin tuo viivettä kommunikaatioon, joten erittäin turvallisuuskriittisten sovellusten yhteydessä siltaratkaisua ei välttämättä voida käyttää. Niinpä etähallintalaitteelta vaaditaan hyvää ja optimoitua suorituskykyä; reitityksen aiheuttamat viiveet väyläliikenteeseen täytyy minimoida, jotta toiminta ei häiriinny.

5.5 Yhteenveto menetelmistä

Kaikki edellisissä luvuissa esitellyt neljä menetelmää täyttävät etähallintalaitteelle asetetut vaatimukset: prosessisignaalien ja hätäviestien kuuntelu sekä palveluobjektioperaatioiden suoritus. Näistä menetelmistä on kuitenkin valittava yksi varsinaista toteutusta varten. Taulukko 5.1 esittää esitellyistä menetelmistä yhteenvedon.

Taulukko 5.1. *Yhteenveto tarkastelluista menetelmistä.*

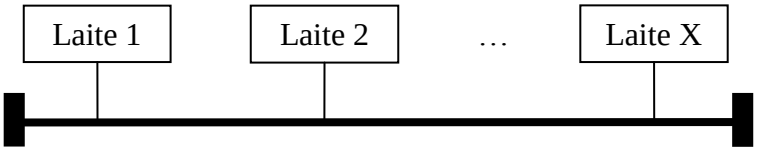
	Kanavien lisääminen	Silta	Hallintalaitteen korvaaminen	Jaettu kanava
Standardi olemassa				x
Vaatii muutoksia alkuperäiseen väylään	x			x
Etähallintalaite yleiskäyttöinen	x	x		x
Rajoittaa CANopen-ominaisuuksia	x			
Vaatii etähallintalaitteelta suorituskykyä		x	x	x
Alkuperäinen hallintalaite huolehtii kriittisistä toiminnoista	x	x		x

Kun tarkastellaan menetelmiä, voidaan huomata, että mikään menetelmistä ei täytä kaikkia taulukossa esitettyjä ominaisuuksia. Näin ollen on valittava tärkeimmät ominaisuudet täyttävä vaihtoehto.

Tärkeimmät menetelmälle asetetut vaatimukset olivat, että etähallintalaite olisi yleiskäyttöinen ja menetelmä ei vaatisi muutoksia alkuperäiseen väylään. Ainoastaan CANopen-silta ja hallintalaitteen korvaaminen etähallintalaitteella eivät vaadi muutoksia alkuperäiseen väylään. Muut menetelmistä vaativat muutoksia joko väylän alkuperäiseen hallintalaitteeseen ja/tai orjalaitteisiin. Näin ollen kyseiset menetelmät on suljettava pois. Hallintalaitteen korvaaminen etähallintalaitteella on suljettava pois siitä syystä, että kyseessä ei ole yleiskäyttöinen ratkaisu, koska etähallintalaitteeseen pitäisi ohjelmoida tiettyyn järjestelmään sidottuja toimintoja, kuten esimerkiksi koneen ohjausalgoritmeja. Näin ollen jäljelle jää ainoataan siltatoteutus, joka valitaan toteutettavaksi menetelmäksi.

6 REITITYS

Perinteisesti CAN-väylät rakennetaan lineaarisen topologian avulla, koska se on ainoa standardoitu väylätopologia. Useissa tapauksissa tämä ei ole verkolle optimaalinen rakenne ja usein kaapelointi on monimutkaisempaa verrattuna esimerkiksi tähtitopologiaan. [26, s. 8.] Lineaarisen väylän periaate on esitetty kuvassa 6.1.

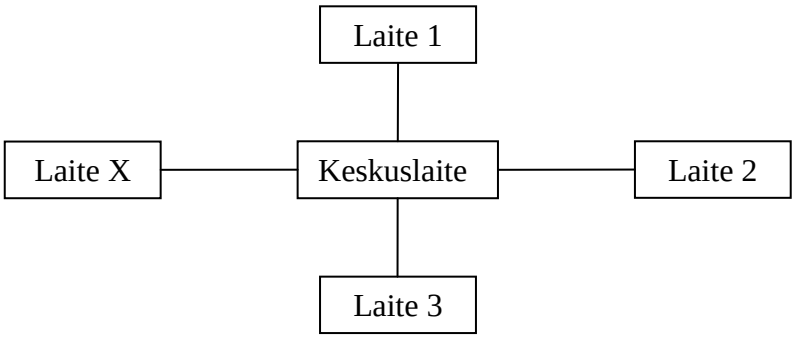


Kuva 6.1. *Lineaarisen väylän periaate.*

Kuvassa laitteet liittyvät samaan fyysiseen kaapeliin. Väylän päässä olevat mustat laatikot kuvaavat väylän päätevastuksia. Lineaarinen väylä rajoittaa väylän laitteiden määrä ja väylän maksimipituus riippuu käytetystä baudinopeudesta [27, s. 1]. Koska kaikki laitteet ovat kiinni samassa fyysisessä väylässä, väylän vikaantuminen aiheuttaa koko CAN-järjestelmän pysähtymisen [28, s. 8].

Monimutkaiset CANopen-järjestelmät koostuvat useista loogisista segmenteistä, joten kaikkia laitteita ei kannata kytkeä samaan lineaariseen väylään. Tällöin voidaan rakentaa hierarkkisia CANopen-verkkoja, joissa loogiset segmentit on jaettu alijärjestelmiksi. Tietyn alijärjestelmän laitteet on kytketty samaan lineaariseen väylään. Saha [29] esittää suunnitteluratkaisut, joiden avulla voidaan toteuttaa erittäin monimutkainen, monitasoinen ja standardia noudattava CANopen-verkko. Periaate on yksinkertainen: jokainen verkkotaso toimii itsenäisesti, mutta ylimmän tason vastuulla on koko verkon konfigurointi ja tilan valvonta, joten eri tasojen väliin tarvitaan CANopen-yhdyskäytävät, jotta kommunikointi tasojen välillä on mahdollista. Näin CANopen-verkosta rakentuu puumainen rakenne. Mikäli jokin taso tai tietyllä tasolla oleva laite vikaantuu, voidaan ylimmällä tasolla päättää, onko kyseessä niin vakava vikaantuminen tai virhetilanne, että koko järjestelmä tulee pysäyttää.

Usein CANopen-järjestelmissä tarvitaan monimutkaisempia väylätopologioita kuin lineaarinen topologia. Näihin lukeutuvat esimerkiksi usein käytetty tähtitopologia, jonka periaate on esitetty kuvassa 6.2.



Kuva 6.2. *CAN-väylän tähtitopologia.*

Kuvasta 6.2 huomataan, että erikoisempien rakenteiden käyttäminen vaatii CAN-väylän osaksi keskuslaitteen, joka mahdollistaa topologian käytön luotettavasti. Näiden laitteiden avulla järjestelmään voidaan myös liittää enemmän laitteita ja väylän maksimipituutta saadaan kasvatettua. Luvut 6.1 ja 6.2 esittelevät CAN-väylän kanssa usein käytetyt keskuslaitteet: keskitin, toistin, kytkin sekä silta.

6.1 Keskitin ja toistin

Keskitin on verkon laite, joka mahdollistaa liikenteen siihen liitettyjen laitteiden välillä. Keskittimessä on useita portteja, joihin laitteita voi liittää ja se voi sekä vastaanottaa että lähettää viestejä, mutta ei samanaikaisesti. Keskitin ei ota kantaa vastaanottamiensa ja lähettämiensä viestien osoitteisiin tai viestien sisältöön. Näin ollen keskitin lähettää kaikki vastaanottamansa viestit kaikkiin keskittimen portteihin muuttumattomina. [30.] Keskitin toimii OSI-referenssimallin fyysisellä tasolla, joten se ei sisällä varsinaisia reititysominaisuuksia. Toistin on keskittimen erityistapaus, joka sisältää ainoastaan kaksi porttia ja toimii keskittimen periaatteiden mukaan.

Saha [26, s. 14] esittää, että keskittimen avulla voidaan toteuttaa CAN-verkkoon kuvassa 6.2 esitetty tähtitopologia. Sahan mukaan keskitin toimii luotettavasti kuitenkin vain 500 kilobaudin tai sitä hitaampien väylänopeuksien kanssa. Joissakin tapauksissa voidaan käyttää myös 800 ja 1000 kilobaudin nopeuksia, mutta tämä rajoittaa tähtitopologian haaran 8,5 ja 2,3 metriin. Näin ollen isompien järjestelmien kanssa keskitintä ei voida käyttää tähtitopologian muodostamiseen. Sahan [27, s. 7] mukaan keskitintä käytettäessä on erittäin tärkeää, että eri haarat on galvaanisesti erotettu toisistaan. Näin mahdollistetaan jokaisessa haarassa mahdollisimman paljon laitteita. Keskitin on kuitenkin hinnaltaan ja toteutukseltaan yksinkertaisempi laite kuin esimerkiksi kytkin, joten hitaampien väylien yhteydessä keskittimien käyttö väylätopologian rakentamiseen on perusteltua.

6.2 Kytkin ja silta

Kytkin mahdollistaa liikenteen siihen liitettyjen laitteiden välillä ja se osaa päätellä, mihin porttiin tietty viesti kuuluu reitittää. Kytkimessä on useita portteja ja se voi vastaanottaa ja lähettää viestejä samanaikaisesti, joten kytkin on keskitintä tehokkaampi

laite. [30.] Viestien reititys suoritetaan reititystaulujen avulla, joiden periaate esitettiin luvussa 5.4 CANopen-sillan yhteydessä. Silta onkin kytkimen erityistapaus, jossa portteja on ainoastaan kaksi. Kytkimet ja sillat kuuluvat OSI-referenssimallin toiselle tasolle eli siirtokerrokselle.

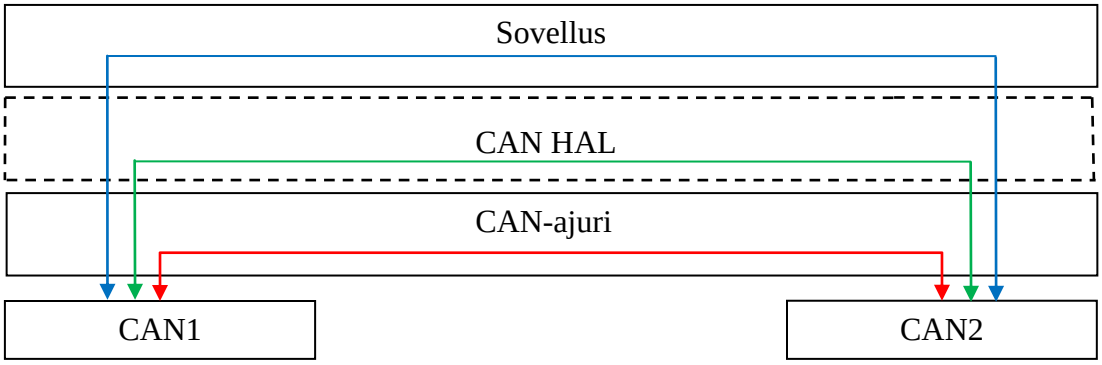
Sahan [27] tutkimuksen perusteella voidaan todeta, että CAN-järjestelmässä kytkimen avulla voidaan luoda mikä tahansa väylätopologia. Kytkimen avulla luodussa verkossa voidaan käyttää myös korkeampia väylänopeuksia ongelmitta toisin kuin keskittimen tapauksessa. Tutkimuksen perusteella kytkimen ja sillan ainoa rajoite on reitityksen aiheuttamat viiveet väylän liikenteeseen. Nämä pitää huomioida, kun harkitaan kytkimen tai sillan käyttöä.

Bäck et al. [28] tutkimus käsittelee CAN-kytkintä, joka on mahdollista konfiguroida käyttäen CANopen-protokollaa. CAN-kytkimen toteutuksessa painotetaan mahdollisimman tehokasta CAN-laitteistoa, jonka avulla reititysviiveet saadaan minimoitua. Tutkimus myös osoittaa, että Ethernet-verkoissa käytetyn viestien puskuroinnin avulla, saadaan myös CAN-kytkimissä tapahtuvat viestien hukkumiset minimoitua. Bäck et al. [28, s. 1] myös esittävät, että CAN-kytkimen käyttö vähentää verkon kuormaa ja tehostaa verkon toimintaa, koska näin pystytään pitämään kommunikointi mahdollisimman tehokkaasti tietyn aliverkon sisällä. Tämä ei ole mahdollista keskittimiä käytettäessä. Tämän lisäksi kytkimiä käytettäessä voidaan käyttää eri väylänopeuksia eri aliverkoissa ja tietyssä aliverkossa tapahtuvat häiriöt viestien ajoituksessa eivät vaikuta koko verkkoon.

Ekiz et al. [25, s. 508–512] esittävät tehokkaan tavan toteuttaa ja suunnitella CAN-silta reititystauluihin pohjautuen. Artikkelissaan he esittävät tarkan algoritmin, jonka mukaan reititys suoritetaan sekä menetelmän CAN-reititystaulujen määrittämiseksi. Artikkelissa esitelty reititysalgoritmi on esitetty liitteessä 1. Reititystaulujen muodostaminen perustuu CAN-protokollan viestien kuittaukseen hyväksyntäbitin avulla (*ACK, acknowledgement*). Kuten aiemmin todettu, aluksi silta reitittää kaikki viestit sillan molemmin puolin. Mikäli viesti kuitataan siltä puolelta siltaa, jonne viesti on reititetty, tiedetään, että viestiä tarvitaan. Kyseinen tieto talletetaan CAN-reititystauluihin ja sitä käytetään jatkossa kyseisen viestin kohdalla.

6.3 CANopen-sillan reititys

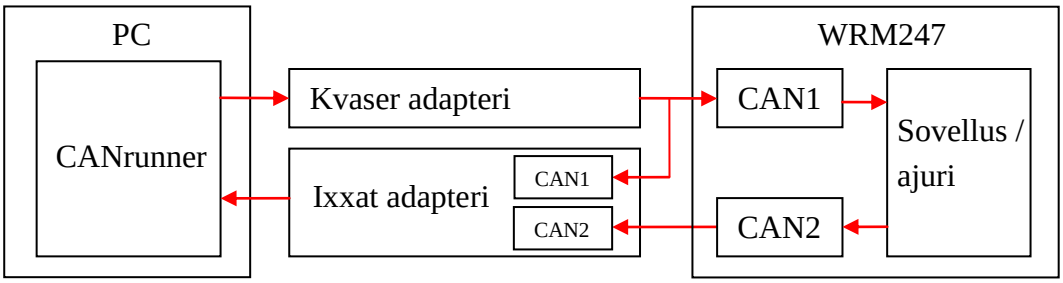
Teoreettisen tarkastelun pohjalta luvussa 5.5 valittiin etähallintalaitteen liittämistavaksi CANopen-silta, jolloin etähallintalaite toimii siltana hallintalaitteen ja orjalaitteiden välillä. Sillantoteutus on huomattavasti yksinkertaisempi, kuin luvussa 6.2 esitellyn reititystauluihin pohjautuvan CAN-sillan, koska reitityksеen liittyy ainoastaan yksi sääntö: etähallintalaitteen palveluobjektioperaatioiden vastauksia ei reititä alkuperäiselle hallintalaitteelle. Näin ollen reititystauluja ei tarvita, mikä johtaa siihen, että reitityksestä tulee huomattavasti kevyempi operaatio, jolloin reitityksen aiheuttamat viiveet pysyvät mahdollisimman alhaisina. Reititys voidaan toteuttaa kolmella eri tasolla: sovellus-, HAL- tai ajuritasolla. Kuva 6.3 havainnollistaa tilanteen.



Kuva 6.3. *Vaihtoehdot reitityksen toteuttamiselle.*

Reitityksessä ei tarvitse huomioida ylemmän tason protokollaa, jos tiedetään, että molemmissa CAN-kanavissa on käytössä sama protokolla. Mikäli kanavissa olisi eri protokolla, täytyisi reititys aina toteuttaa sovellus- tai HAL-tasolla, koska CAN-kehyksille pitäisi suorittaa muunnos protokollasta toiseen. HAL- ja sovellustason reitityksen toteutuksilla ei ole käytännössä mitään eroa. HAL-taso halutaan kuitenkin pitää mahdollisimman riippumattomana tiettyyn sovellukseen sidotuista toiminnallisuuksista, joten HAL-tason reititys suljetaan tästä syystä vertailuista pois. Näin ollen vertailu suoritetaan kahden toteutusmahdollisuuden välillä: sovellus- ja ajuritaso.

Jotta reitityksen toteutustapoja voidaan luotettavasti vertailla, tulee määritellä testijärjestely, jolla voidaan varmentaa erot toteutustapojen välillä. Kuva 6.4 esittää testijärjestelyn, jolla määritettiin reitityksen aiheuttama viive. Kuvassa viestien kulkusuunta on merkitty punaisella.



Kuva 6.4. *Reititysviiveiden mittaus.*

Testissä diagnostiikkaohjelmana käytetään Wapicella kehitettyä CANrunner-työkalua, joka mahdollistaa kahden tai useamman CAN-adapterin käytön samanaikaisesta. Väyläkuorman tuottamiseen käytetään Kvaserin valmistamaa Kvasef Leaf Semi Pro HS USB-CAN -adapteria ja viestien vastaanottoon käytetään Ixxatin kaksikanavaista USB-to-CAN-2 -adapteria. Kuvassa Ixxatin adapterin CAN1-kanava kytketään Kvaserin adapterin ja WRM247-laitteen CAN1-kanavan kanssa samaan CAN-väylään. Ixxatin adapterin CAN2-kanava puolestaan kytketään WRM247-laitteen CAN2-kanavan kanssa samaan CAN-väylään. Näin voidaan määrittää, kuinka paljon nopeammin viesti on saa-

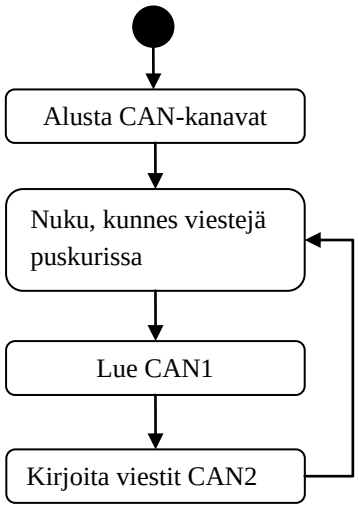
tavilla Ixxatin adapterin CAN1-kanavassa verrattuna Ixxatin adapterin CAN2-kanavaan. Kuvassa reititys tapahtuu joko sovellus- tai ajuritasolla.

Testeissä määritetään reitityksen aiheuttamat viiveet sekä prosessorikuormat kahdelle valitulle toteutusmahdollisuudelle. Jokaisessa testimittauksessa lähetetään 500000 CAN-viestiä, joiden tietosisältö on kahdeksan tavua. Tietosisältönä käytetään koko ajan kasvavaa laskuria, jolloin viestit on helppo tunnistaa viiveiden laskemista varten. Testeihin valittiin väylänopeudeksi 500 kilobaudia.

6.3.1 Reititys sovellustason kautta

WRM-etähallintalaitteessa on Linux käyttöjärjestelmä, joten sovellustason reititys tapahtuu käyttäjätilassa (*user space*). Niinpä lähtökohtaisesti reititys sovellustason kautta on hitaampaa kuin reititys ajuritasolla, mutta reitityksen toteuttaminen sovellustasolla on erittäin yksinkertaista ja tällöin ei myöskään tarvitse tehdä muutoksia ajureihin. Vaikka reititys ajuritasolla ei olisikaan käytössä, tulee silti ajureiden viestien vastaanottofunktioon vertailuja lisää, joten normaali toiminta hidastuisi muutamia kellojaksoja. Tämä tulee ottaa huomioon, koska CAN-ajureita käyttävät useat muut ohjelmiston osat. Lisäksi sovellustason reitityksen testaus ja todentaminen on helppoa, mutta ajuritason testaus vaatii huomattavasti enemmän tarkkuutta ja työtunteja.

Sovellustason reitityksen viiveiden mittauksessa käytetään yksinkertaisinta mahdollista tilannetta, joka on esitetty kuvassa 6.5.



Kuva 6.5. *Sovellustason reitityksen viivemittaus.*

Sovellus lukee CAN-viestit CAN1-liitynnästä ja kaiuttaa luetut viestit CAN2-liityntään. Sovellus nukkuu aina, kun viestejä ei ole saatavilla, koska muuten prosessorikuorma olisi 100 %. Saadut tulokset on esitetty taulukossa 6.1.

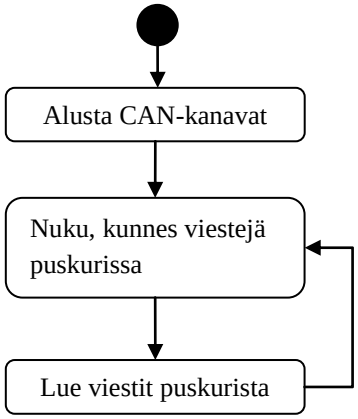
Taulukko 6.1. *Sovellustason reitityksen aiheuttamat viiveet ja prosessikuormat.*

Viestinopeus (viestiä/s)	Minimiviive (us)	Keskiarvoviive (us)	Maksimiviive (us)	Prosessorikuorma (%)
250	474	1823	35466	16
500	486	1929	37433	28
750	416	2049	35227	27
1000	487	2216	36697	38
1250	376	2479	40776	38
1500	520	2479	37013	46
1750	458	2019	34697	51
2000	521	3000	37037	54

Kuten taulukosta 6.1 voidaan huomata, sovellustasolla reitityksen maksimiviiveet kasvavat melko suuriksi, vaikka minimi- ja keskiarvoviiveet ovatkin suhteellisen pieniä. Maksimiviive nousee pahimmassa tapauksessa yli 40 millisekunnin. Maksimiviive on viiveistä kaikista tärkein, koska sen avulla voidaan määrittää pahin mahdollinen tapaus eli aika, jota suuremmaksi viestin reititysviive ei varmasti kasva. Myös prosessorikuormat ovat suuria, kun otetaan huomioon, että kyseessä on ainoastaan viestien reititys eli muuta sovellustason toiminnallisuutta ei ole vielä mukana. Taulukossa esitetyt prosessorikuormat johtuvat ainoastaan viestien reitityksestä.

6.3.2 Reititys ajuritason kautta

Ajuritasolla toteutettuna sovellustaso on erittäin yksinkertainen. Se on esitetty kuvassa 6.6.

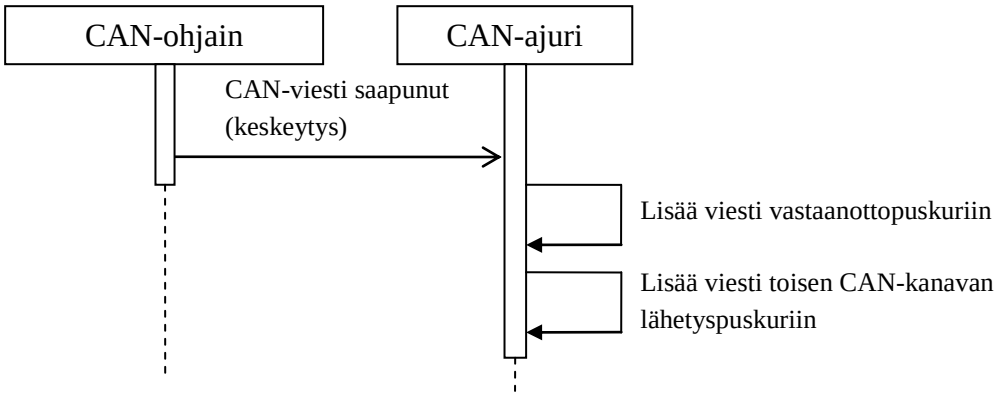


Kuva 6.6. *Ajuritasolla toteutetun reitityksen sovellustason tilakaavio.*

Sovelluksen tehtävä on ainoastaan alustaa CAN-liitynnät ja lukea saapuneet viestit pois CAN1-kanavan puskurista, jotta puskuri ei vuoda yli. Viestejä odotettaessa sovellustaso nukkuu, koska muuten prosessorikuorma olisi 100 %. Viestejä ei välttämättä tarvitsisi lukea pois CAN-kanavan puskurista sovellustasolla, jolloin taulukossa 6.2 esitetyt pro-

sessorikuormat johtuisivat pelkästään ajuritason reitityksestä. Viestien luku kuitenkin haluttiin ottaa mukaan, koska näin tilanne vastaa paremmin todellisuutta.

Kun reititys toteutetaan ajuritasolla, ei se näy mitenkään sovellustasolla. Ajurit ovat osa WRM-etähallintalaitteen Linux-ydintä, joten ajuritason reititys tapahtuu ytimen sisällä (*kernel space*). Ajuritason reitityksen toiminta on esitetty kuvassa 6.7.



Kuva 6.7. *Ajuritason reitityksen toimintaperiaate.*

Kuvassa CAN-ohjain aiheuttaa keskeytyksen aina, kun se vastaanottaa viestin CAN-väylältä. Vastaanottokeskeytys käsitellään CAN-ajurin keskeytyskäsittelijässä, joka tutkii, kumman CAN-kanavan vastaanottokeskeytys on kyseessä. Tämän jälkeen ajuri lisää vastaanotetun viestin kyseisen kanavan vastaanottopuskuriin sekä kopioi viestin toisen CAN-kanavan lähetyspuskuriin. Näin toteutettuna reititys on mahdollisimman nopeaa ja viiveiden pitäisi olla huomattavasti pienempiä kuin sovellustason reitityksessä. Reititysviiveet ja prosessorikuormat ajuritasolla on esitetty taulukossa 6.2.

Taulukko 6.2. *Ajuritason reitityksen aiheuttamat viiveet ja prosessikuormat.*

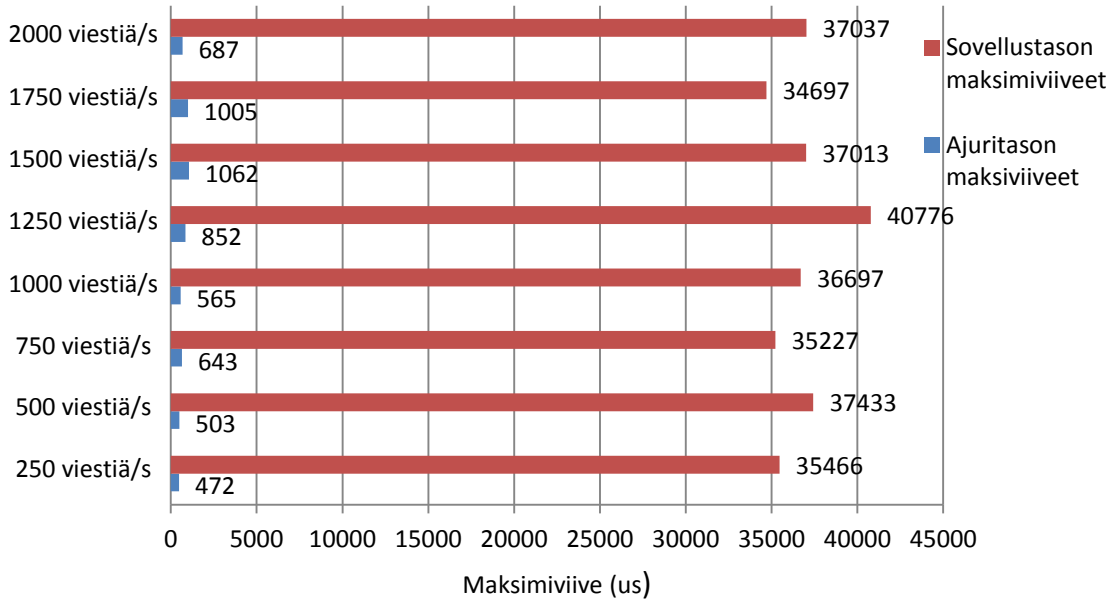
Viestinopeus (viestiä/s)	Minimiviive (us)	Keskiarvoviive (us)	Maksimiviive (us)	Prosessorikuorma (%)
250	169	318	472	12
500	157	317	503	20
750	158	369	643	22
1000	136	320	565	32
1250	183	422	852	32
1500	183	438	1062	38
1750	160	465	1005	44
2000	140	340	687	51

Ajuritasolla reitityksen minimi-, keskiarvo- ja maksimiviiveet ovat hyvin lähellä toisiaan eli hajonta on pientä. Pahimmassakin tapauksessa maksimiviive nousee yhden millisekunnin tasolle. Prosessorikuormat nousevat myös ajuritasolla suuriksi, mikä saattaa tuottaa ongelmia lopullisessa sovelluksessa. Taulukossa esitetyt prosessorikuormat eivät johdu kokonaan ajuritason reitityksestä, vaan prosessorikuormissa tulee ottaa huomioon,

että myös kuvassa 6.6 esitetty sovellustasolla tapahtuva viestien lukeminen lisää prosessorikuormaa.

6.3.3 Reititysmittausten tulokset

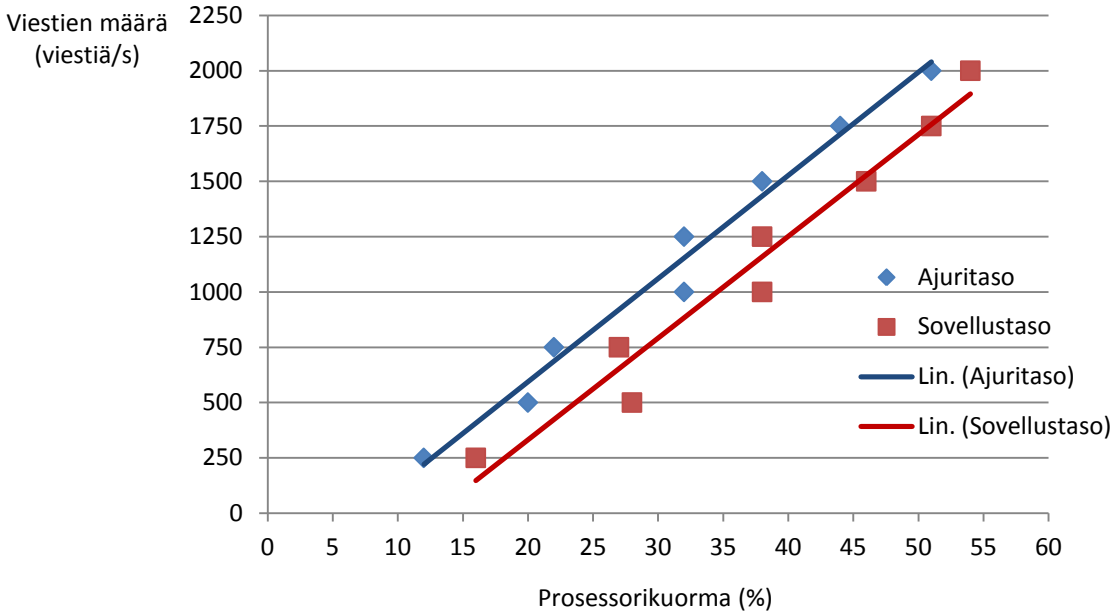
Kun vertaillaan sovellus- ja ajuritason reititysviivemittauksista saatuja tuloksia, huomataan, että ajuritasolla toteutettuna reitityksen viiveet jäävät huomattavasti alhaisemmiksi kuin sovellustasolla. Kuva 6.8 havainnollistaa reitityksen maksimiviiveiden erot eri viestinopeuksilla.



Kuva 6.8. *Reitityksen tuottamat maksimiviiveet.*

Kuvasta nähdään, että ajuritasolla suurin maksimiviive on 1062 mikrosekuntia ja sovellustasolla 40776 mikrosekuntia. Kun kuvasta lasketaan keskiarvo kaikkien viestinopeuksien sovellus- ja ajuritason välillä, on sovellustason maksimiviive keskimäärin 55-kertainen verrattuna ajuritasoon. Tämä on merkittävä ero, koska maksimiviiveen avulla määritetään pahin mahdollinen reitityksen aiheuttama viive. Taulukoita 6.1 ja 6.2 vertailemalla huomataan, että myös minimi- ja keskiarvoviiveet ovat sovellustasolla moninkertaisia verrattuna ajuritasoon.

Viestien reitityksen aiheuttamat prosessorikuormat puolestaan ovat suhteellisen lähellä toisiaan sovellus- ja ajuritasolla. Kuva 6.9 esittää aiheutuneet prosessorikuormat sekä prosessorikuormien lineaarisoinnin.



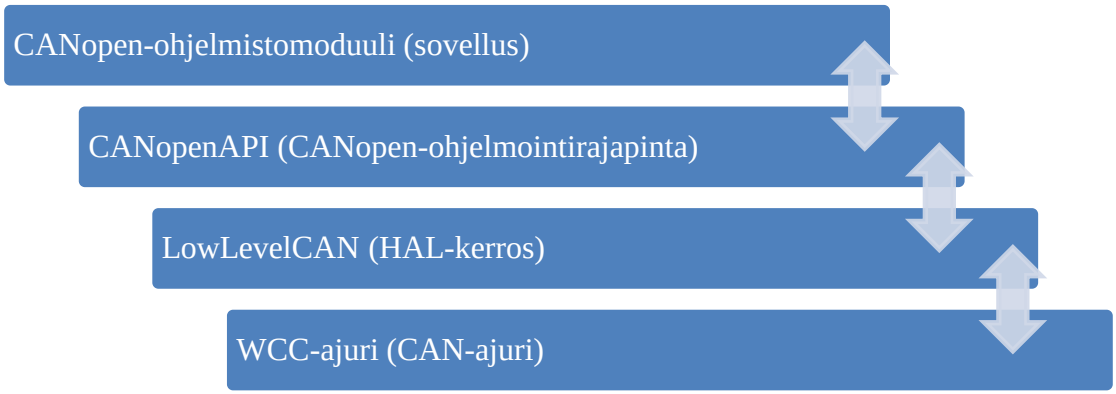
Kuva 6.9. *Reitityksen tuottamat prosessorikuormat.*

Kuvasta laskettuna lineaarisoinnin tuottamien suorien ero on 5,5 %. Näin ollen ajuritasolla reitityksestä aiheutuva prosessorikuorma on keskimäärin 5,5 % pienempi kuin sovellustasolla. Vaikka ajuritason prosessorikuorma on pienempi, ei ero ole yhtä merkittävä kuin reititysviiveiden tapauksessa. Suuri osa prosessorikuormasta aiheutuu CAN-liikenteen aiheuttamista keskeytyksistä ja kontekstinvaihdoista Linuxin käyttäjätilan ja ytimen välillä. Prosessorikuormat ovat lähellä toisiaan, koska sekä ajuri- että sovellustasolla keskeytyksiä ja kontekstinvaihtoja on lähes yhtä paljon.

Kun otetaan huomioon sekä reititysviiveet että prosessorikuormat, voidaan todeta, että ainakin kriittisten viestien reititys tulisi toteuttaa ajuritasolla. Mikäli viestit eivät ole järjestelmän toiminnan kannalta kriittisiä tai niiden prioriteetti ei ole korkea, voidaan ne reitittää sovellustason kautta.

7 SUUNNITTELURATKAISUT

Valittu siltatoteutus mahdollistaa erittäin yksinkertaisen ohjelmistorakenteen, koska väylän alkuperäinen hallintalaite hoitaa verkon käynnistyksen, hätäviestien edellyttämät toimenpiteet ja muut hallintatoimenpiteet, joten etähallintalaitteen ei tarvitse toteuttaa kyseisiä toimintoja. Valittu ohjelmistorakenne on esitetty kuvassa 7.1.



Kuva 7.1. *WRM247-etähallintalaitteen CANopen-ohjelmistorakenne.*

Työssä kehitettiin kokonaan alusta lähtien CANopen-ohjelmistomoduuli, joka käyttää hyväkseen työssä kehitettyä CANopen-ohjelmointirajapintaa. HAL-kerrokseksi valittiin Wapicella kehitetty LowLevelCAN, jota on käytetty useissa ohjelmistoprojekteissa ja jonka toiminta on todettu hyvin luotettavaksi ja tehokkaaksi. Ajureiksi valittiin Wapicella kehitetty WCC-ajuri (*Wapice Custom CAN*), joka on optimoitu huomattavasti suorituskykyisemmäksi kuin perinteiset Linux-järjestelmissä käytetyt CAN-ajurit [31]. Ohjelmistorakenne on valittu siten, että se olisi mahdollisimman suorituskykyinen, koska luvussa 6 esitettyjen mittaustulosten perusteella viestien reititys vie suuren osan laitteen suorituskyvystä.

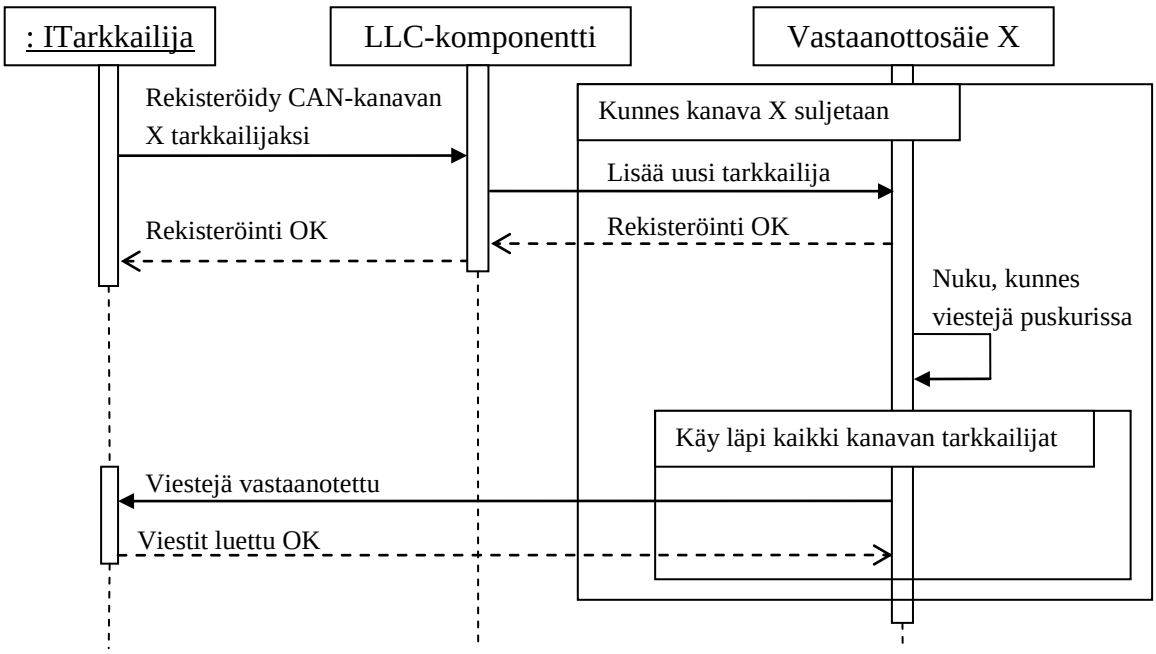
Kuten rakenteesta huomataan, CANopen-ohjelmistopino on jätetty kokonaan pois. Ohjelmistopinon käyttö lisäisi viivettä ja ohjelmakoodin kokoa huomattavasti ja sen käyttäminen vaatisi paljon enemmän suorituskykyä. Ohjelmistopinon pois jättäminen on perustelua, koska CANopen-operaatioista tarvitaan ainoastaan palveluobjektien-luku- ja kirjoitus, jotka voidaan toteuttaa suoraan CANopen-ohjelmointirajapintaan. Siltatoteutus ei vaadi laitteelta omaa objektikirjastoa, koska etähallintalaite ei koskaan toimi orjalaitteena. Se ei lähetä väylälle omia mittauksiaan, tilatietojaan tai heartbeat-viestejä ja se ei voi toimia palveluobjektipalvelimena. Mikään väylän laitteista ei tiedä, että etähallintalaite on kiinni väylässä. Etähallintalaitteen tehtävä on kuunnella väylää, reitittää väylän viestit ja tehdä omia palveluobjektikyselyjään.

7.1 CAN-ajurit ja HAL-kerros

CAN-ajureiksi valittiin Wapicella toteutettu WCC-ajuri. Ajurit on optimoitu suorituskyvyn suhteen, joten ne sopivat erinomaisesti käytettäväksi CANopen-siltatoteutuksen kanssa. Junnila et al. [31, s. 1-9] esittävät artikkelissaan periaatteet, joiden avulla WCC-ajurit on optimoitu. Näihin optimointeihin kuuluvut esimerkiksi esiprosessorihaaravihjeet (*engl. brach hints*), *inline*-määreiden käyttö ja tiedon oikosiirto (*engl. DMA, Direct Memory Access*). Artikkelissaan Junnila et al. [31, s. 1-9] esittävät useita suorituskykymittauksia ja vertailuja muihin perinteisiin Linux-järjestelmässä käytettyihin CAN-ajureihin. Lopputuloksena voidaan todeta, että WCC-ajurit ovat suorituskyvyltään huomattavasti paremmalla tasolla kuin perinteisesti Linux-järjestelmien kanssa käytetyt SocketCAN- ja LinCAN-ajurit.

Ajurit abstrahoidaan HAL-kerroksen avulla. Työssä valittiin käytettäväksi Wapicella kehitetty LLC-abstrahointikomponentti (*LowLevelCAN*). Kyseinen komponentti tarjoaa tarvittavat palvelut CAN-kanavien alustamiseen, sulkemiseen ja parametrien muuttamiseen sekä viestien lukemiseen ja kirjoittamiseen. LLC-komponentti noudattaa singleton-suunnittelumallia, jossa komponentista voidaan luoda koko järjestelmässä ainoastaan yksi instanssi. LLC-komponentti hallitsee koko järjestelmän CAN-kanavia ja pitää kirjaa niin auki olevista kanavista kuin kanavien parametreistakin.

LLC-komponentti käyttää hyväkseen viestien vastaanotossa tarkkailija-suunnittelumallia (*Observer Design Pattern*). Tarkkailija-suunnittelimallissa on kaksi pääroolia: tarkkailija ja kohde. Kohde sisältää tarkkailijalle merkityksellisen tapahtuman tai toiminnon. Tarkkailija rekisteröi itsensä kohteelle ja kun tapahtuma aktivoituu, ilmoittaa kohde tästä kaikille rekisteröityneille tarkkailijoille. [32, s. 370–370.] LLC-komponentin tarkkailija-malli on esitetty kuvassa 7.2.



Kuva 7.2. *LLC-komponentin tarkkailija-suunnittelumalli.*

Kuvassa esitetyllä sekvenssikaaviolla esiehtona on, että CAN-kanava X on valmiiksi avattu. Mikäli kanavaa ei ole avattu, tarkkailijaksi rekisteröityminen ei onnistu. Kuvassa esitetty vastaanottosäie luodaan CAN-kanavan avauksen yhteydessä automaattisesti. Sekvenssikaavion alussa *ITarkkailija*-rajapinnan toteuttava olio rekisteröi itsensä LLC-komponentille tietyn CAN-kanavan kuuntelijaksi. LLC-komponentti välittää referenssin tarkkailijaan kyseisen kanava vastaanottosäikeelle. Kyseinen säie nukkuu, kunnes avatusta CAN-kanavasta voidaan vastaanottaa viestejä. Kun viestit on luettu ajureilta, kutsuu CAN-kanavan vastaanottosäie kyseiselle kanavalle rekisteröityneiden tarkkailijoiden CAN-viestien vastaanottopalvelua, jonka parametrina on osoitin vastaanotettuihin viesteihin. Näin toteutettuna viestit vastaanotetaan asynkronisesti ja tehokkaasti, eikä tarkkailijoiden tarvitse erikseen kutsua LLC-komponentin viestien lukupalvelua.

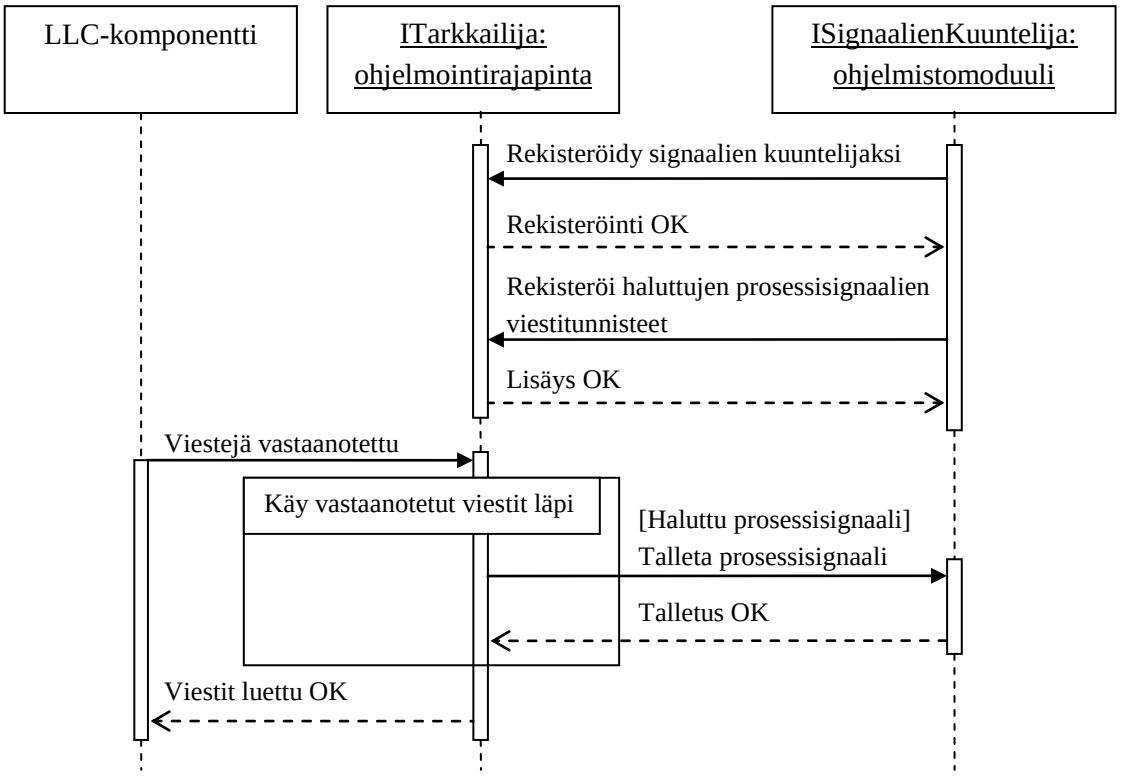
7.2 CANopen-ohjelmistomoduuli ja -ohjelmointirajapinta

CANopen-ohjelmistomoduuli ja -ohjelmointirajapinta kehitettiin kokonaan tässä työssä. Ne pyrittiin suunnittelemaan rajapintoja käyttämällä, jotta niiden toteutukset eivät olisi sidottuja toisiinsa. CANopen-ohjelmistomoduulin tehtävä on suorittaa CAN-kanavien alustaminen annettujen parametrien mukaan, suorittaa palveluobjektioperaatiot ohjelmointirajapinnan palveluiden avulla sekä toteuttaa palvelut, joiden avulla voidaan tallettaa halutut hätäviestit ja prosessisignaalit levylle. Ohjelmistomoduulin konfigurointi tapahtuu WRM-palvelimen avulla, josta voidaan esimerkiksi valita prosessisignaalit, joita kuunnellaan sekä käytetty baudinopeus. Lisäksi palvelimen avulla voidaan käskeä WRM-laite suorittamaan palveluobjektioperaatioita.

CANopen-ohjelmointirajapinta tarjoaa tarvittavat palvelut ohjelmistomoduulille. Se tarjoaa muun muassa palvelut palveluobjektien lukuun ja kirjoittamiseen sekä mahdollisuuden rekisteröityä prosessisignaalien ja hätäviestien kuuntelijaksi. Prosessisignaaleille on myös mahdollista määrittää, mitä prosessisignaaliviestitunnisteita halutaan vastaanottaa. CANopen-ohjelmointirajapinnan instanssi toteuttaa luvussa 7.1 esitellyn *ITarkkailija*-rajapinnan ja kyseinen instanssi rekisteröidään LLC-komponentille vastaanotettujen viestien kuuntelijaksi. Ohjelmointirajapinnan toteuttava olio pitää myös kirjaa väylän alkuperäisen hallintalaitteen käynnissä olevista palveluobjektioperaatioista. Tätä tietoa tarvitaan, kun etähallintalaite haluaa tehdä omia palveluobjektioperaatioitaan.

7.2.1 Prosessisignaalien ja hätäviestien tarkkailu

CANopen-ohjelmistomoduulin ja -ohjelmointirajapinnan tärkeimpiä tehtäviä on hoitaa väylällä liikkuvien prosessisignaalien ja hätäviestien tarkkailu. Kuva 7.3 esittää, miten prosessisignaaleita tarkkaillaan.



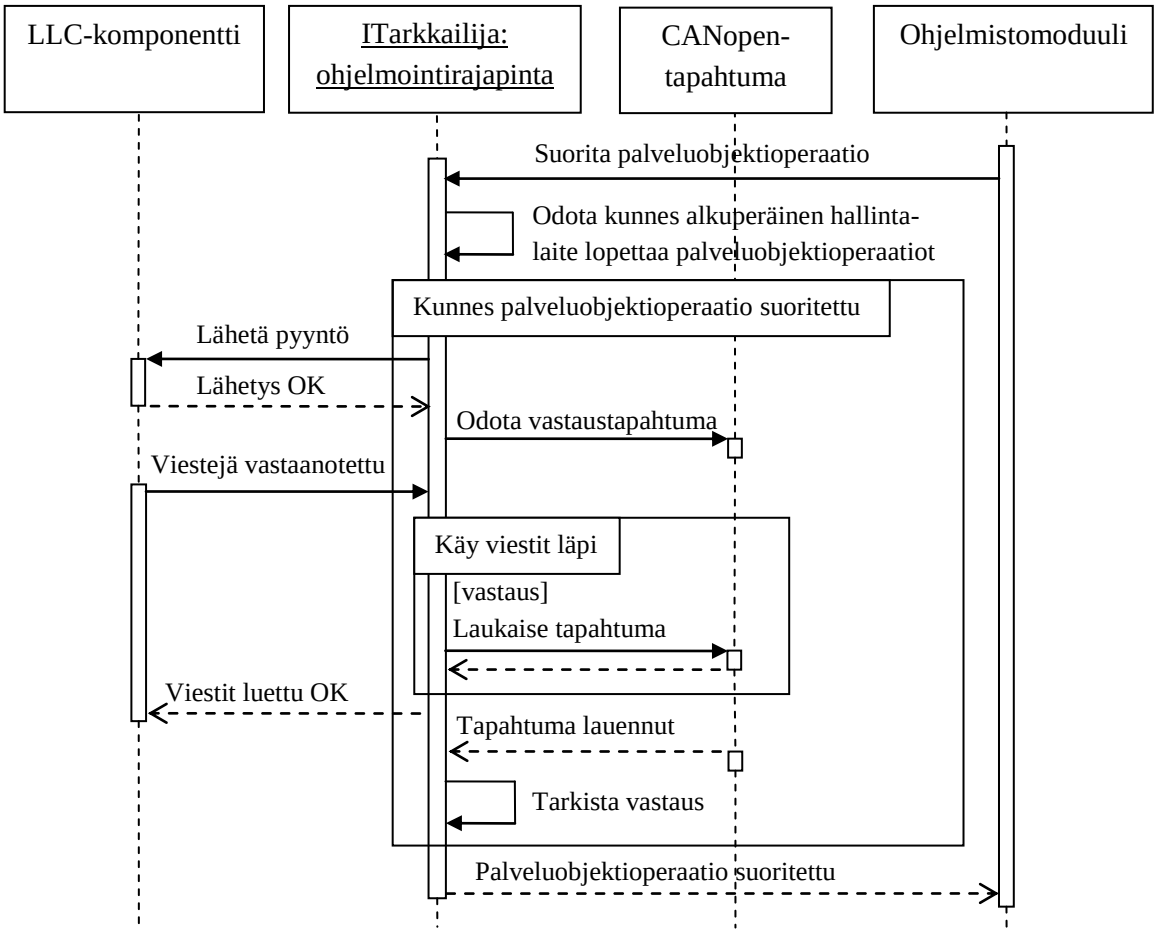
Kuva 7.3. *Prosessisignaalien vastaanotto.*

CANopen-ohjelmistomoduuli toteuttaa *ISignaalienKuuntelija*-rajapinnan ja rekisteröi itsensä CANopen-ohjelmointirajapinnan toteuttavalle oliolle signaalien kuuntelijaksi. Kun rekisteröinti on suoritettu, ohjelmistomoduuli voi rekisteröidä niiden prosessisignaalien viestitunnisteet, joita se haluaa kuunnella. Kun LLC-komponentti kutsuu ohjelmointirajapinnan instanssin viestienvastaanotto funktiota, tarkistetaan, onko jokin viesteistä haluttu prosessisignaaliviesti. Mikäli on, kutsutaan signaalin kuuntelijan prosessisignaalin tallennuspalvelua.

Hätäviestit vastaanotetaan samalla tavalla kuin prosessisignaalit kuvassa 7.3. *ISignaalienKuuntelija*-rajapinta sisältää myös palvelun hätäviestien talletukseen. Ainoa ero prosessisignaalien ja hätäviestien välillä on, että haluttuja hätäviestien viestitunnisteita ei tarvitse erikseen rekisteröidä ohjelmointirajapinnan instanssille. CANopen-protokolla määrittelee hätäviestien viestitunnisteille tietyn viestitunnisteavaruuden ja kaikki tähän avaruuteen kuuluvat viestit halutaan tallettaa, koska hätäviestit ovat järjestelmän kannalta kriittisiä.

7.2.2 Palveluobjektioperaatiot

Palveluobjektioperaatioiden suorittaminen on etähallintalaitteen toinen tärkeä tehtävä. Kuva 7.4 esittää, kuinka palveluobjektioperaatiot suunniteltiin.



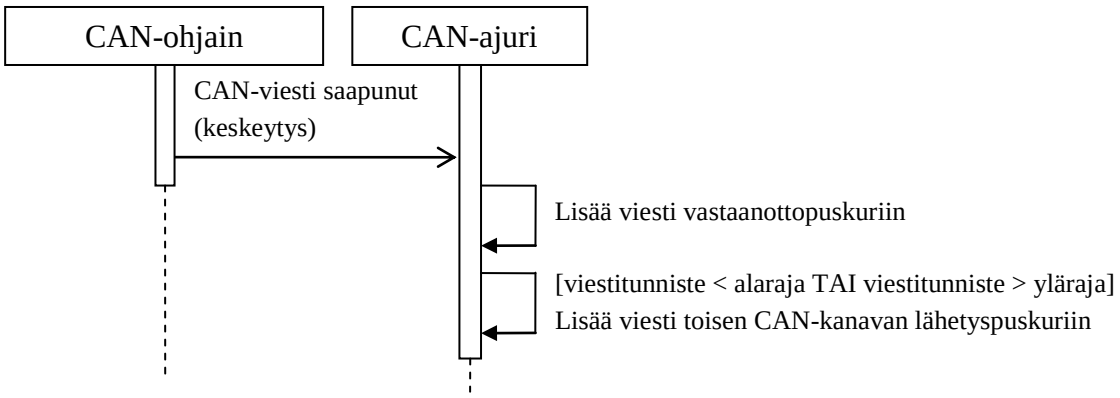
Kuva 7.4. *Palveluobjektioperaation suoritus.*

Kuvassa 7.4 suoritettava palveluobjektioperaatio on joko luku- tai kirjoitusoperaatio. CANopen-ohjelmistomoduuli kutsuu CANopen-ohjelmointirajapinnan palvelua (luku- tai kirjoituspalvelu). Tämän jälkeen odotetaan, että väylän alkuperäisellä hallintalaitteella ei ole palveluobjektioperaatioita käynnissä. Näin varmistetaan, että etähallintalaite ei häiritse alkuperäisen väylän toimintaa. Kun odotus on päättynyt, lähetetään palveluobjektipyyntö halutulle laitteelle LLC-komponentin palveluiden avulla, minkä jälkeen siirrytään odottamaan CANopen-vastaustapahtumaa. Kun viestejä vastaanotetaan väylältä, kutsutaan LLC-komponentin sisäisestä vastaanotto säikeestä CANopen-ohjelmointirajapinnan viestien vastaanottopalvelua. Vastaanottopalvelussa käydään kaikki vastaanotetut viestit läpi ja mikäli jokin viesteistä on vastaus lähetettyyn palveluobjektipyyntöön, laukaistaan vastaustapahtuma. Kun huomataan, että vastaustapahtuma on lauennut, tarkistetaan saatu vastaus. Tämän jälkeen palveluobjektioperaatiota jatketaan samalla syklillä, kunnes operaatio on saatu valmiiksi. Lopuksi palautetaan saatu vastaus CANopen-ohjelmistomoduulille.

7.3 Reititys lopullisessa sovelluksessa

Lopullisessa sovelluksessa reititys toteutetaan kahdella tasolla: sovellus- ja ajuritasolla. Kuten aiemmin todettu sovellustason reitityksen viiveet kasvavat kriittisten viestien kohdalla liian suuriksi, joten parhaassa mahdollisessa tapauksessa kaikki viestit reititettäisiin ajuritasolla. Se ei kuitenkaan ole mahdollista, koska palveluobjektien reititys vaatii sovellustason logiikkaa eli palveluobjektiviestit on pakko reitittää sovellustason kautta. Ajureissa ei voida tietää, onko orjalaitteilta tuleva palveluobjektiviesti vastaus väylän hallintalaitteen vai etähallintalaitteen tekemään palveluobjektipyyntöön. Mikäli viesti on vastaus etähallintalaitteen tekemään pyyntöön, sitä ei saa reitittää väylän hallintalaitteelle. Tämä päättely vaatii tarkkaa kirjanpitoa käynnissä olevista palveluobjektioperaatioista ja se täytyy toteuttaa sovellustasolla. Hallintalaitteelta tulevat palveluobjektipyynnöt pitää reitittää sovellustason kautta, koska muuten etähallintalaite ja väylän hallintalaite saattavat tehdä samanaikaisesti palveluobjektioperaatioita samalle orjalaitteelle. Tämä tilanne voidaan pois sulkea sovellustasolla. Järjestelmän toiminnan kannalta palveluobjektit ovat kuitenkin matalan prioriteetin viestejä eli ne voidaan reitittää sovellustason kautta.

Ajuritasolla reitite tään kaikki muut viestit paitsi palveluobjektiviestit. Ongelma on kuitenkin se, että miten ajureille voidaan helposti kertoa, mitkä viestit tulee reitittää? CANopen-viestitunnisteet rakentuvat siten, että tämä ei ole mahdollista yksinkertaisella viestitunnistemaskilla. Kuten luvussa 3.4 todettiin, CANopen-protokollassa palveluobjekteille on varattu tietty, yhtenäinen alue viestitunnisteavaruudesta. Tätä tietoa voidaan käyttää hyväksi viestien reitityksessä. Kuva 7.5 havainnollistaa tilanteen.



Kuva 7.5. *Ajuritason reitityksen lopullinen toteutus.*

Kuvassa 7.5 on sama tilanne kuin kuvassa 6.7, mutta nyt viesti lisätään toisen CAN-kanavan lähetyspuskuriin vain, jos viestitunniste on pienempi kuin määritelty alaraja tai suurempi kuin määritelty yläraja. Ala- ja ylärajoiksi molemmille WRM-laitteen CAN-liitynnöille määritellään palveluobjekteille varatun viestitunnisteavaruuden ala- ja yläraja. Jotta ratkaisu olisi mahdollisimman yleiskäyttöinen, määriteltiin ajureiden *ioctl*-funktioon uusia pyyntökoodeja. Kyseisen funktion avulla voidaan manipuloida Linux-järjestelmässä tiettyyn oheislaitteeseen sidottuja toimintoja ja parametreja sekä uusia

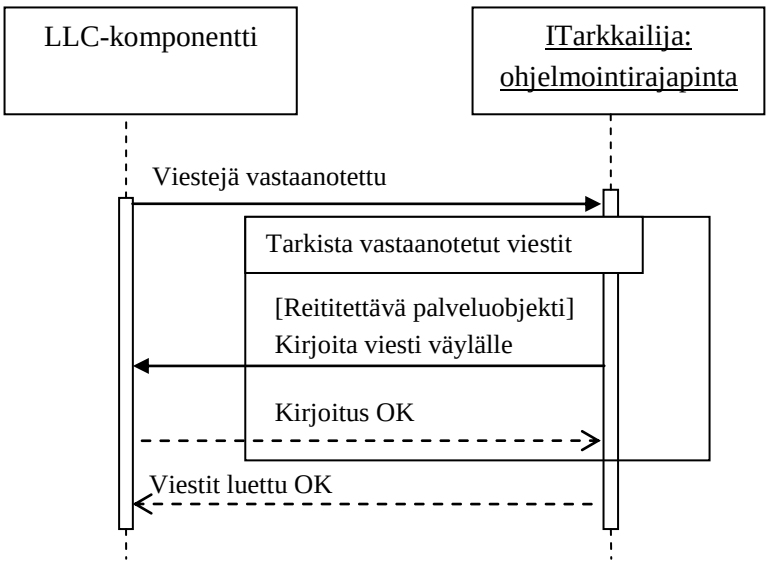
pyyntökoodeja määrittelemällä voidaan laitteeseen lisätä uusia toiminnallisuuksia [33]. Taulukossa 7.1 on esitetty pyyntökoodit reitityksen toteuttamiseksi.

Taulukko 7.1. *Ajureihin määritellyt pyyntökoodit reititystä varten*

Pyyntökoodi	**Merkitys**
ROUTING_LIMIT_HIGH	Asettaa reitityksen ylärajan
ROUTING_LIMIT_LOW	Asettaa reitityksen alarajan
ENABLE_ROUTING_LIMITS	Ottaa käyttöön sekä ylä- että alarajan
DISABLE_ROUTING_LIMITS	Poistaa käytöstä sekä ylä- että alarajan

Taulukosta havaitaan, että tarvitaan yhteensä neljä uutta pyyntökoodia. Kuten huomataan, reititysrajat voidaan asettaa ajureille, joten rajoja voidaan sovelluksen mukaan muuttaa tarvittaessa. Rajat tulee ottaa käyttöön erillisellä pyyntökoodilla, koska muuten ei voida varmistaa, että asetetut ylä- ja alaraja otetaan käyttöön ajureiden näkökulmasta atomisesti. Kun rajoja ei enää haluta käyttää, otetaan ne pois käytöstä myös erillisellä pyyntökoodilla.

Sovellustasolla reititetään palveluobjektiviestit. Reititys tapahtuu kuvan 7.6 mukaisesti.



Kuva 7.6. *Palveluobjektien reititys.*

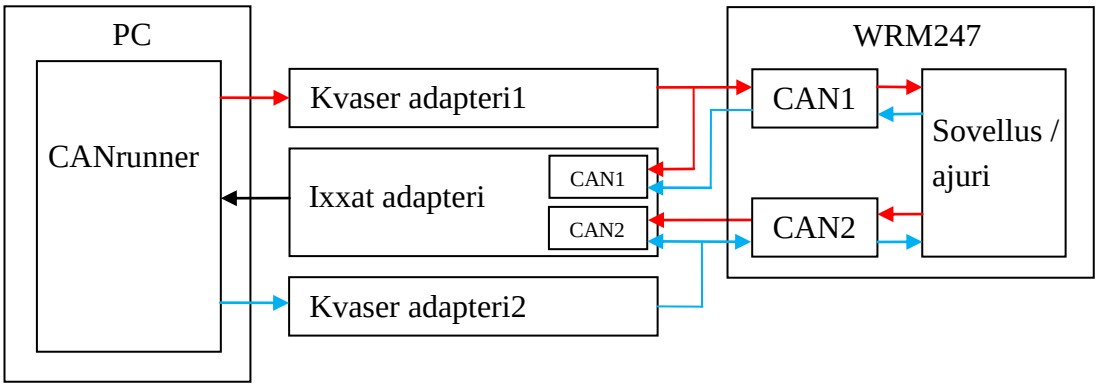
Kuvassa LLC-komponentti kutsuu *ITarkkailija*-rajapinnan toteuttavaa CANopen-ohjelmointirajapinnan instanssia ja sen viestien vastaanottopalvelua. Ohjelmointirajapinta tarkistaa kaikki vastaanotetut viestit ja jos jokin niistä on palveluobjekti viesti, joka pitää reitittää, se kutsuu LLC-komponentin viestien kirjoituspalvelua kyseiselle viestille. Viesti kirjoitetaan eri CAN-kanavaan, kuin mistä viesti vastaanotettiin. Ohjelmointirajapinnan instanssi myös pitää kirjaa väylän alkuperäisen hallintalaitteen tämän hetkisistä palveluobjektioperaatioista.

8 TOTEUTUKSEN TUNNUSLUKUJA

Luvussa 7 esiteltyjen periaatteiden mukaan toteutetun CANopen-sillan avulla mahdollistetaan tavoitellut prosessisignaalien ja hätäviestien kuuntelu sekä palveluobjektioperaatioiden suoritus. Nämä ominaisuudet testattiin huolellisesti myös mahdollisissa virhetilanteissa useiden kaupallisten laitteiden avulla. Tärkein osa toteutuksen testausta kuitenkin on, että väylän alkuperäinen toiminta ei häiriinny reitityksen seurauksena. Käytännössä tämä tarkoittaa, että sillan aiheuttamat viiveet on pidettävä lopullisessa sovelluksessa mahdollisimman alhaisina.

On vaikea määrittää yleispäteviä rajoja, jotka ilmaisevat reititykselle sallitut viiveet. Kyseiset rajat riippuvat tarkasteltavasta järjestelmästä. Ekiz et al. [34, s. 185–187] esittävät SAE-suorituskykytestin käyttöä siltaratkaisun reititysviiveiden määrityksessä. Mittauksessa käytetään 53 eri viestitunnistetta, jotka jaetaan yhteensä seitsemän eri lähettävän laitteen kesken. Jokaiselle viestille määritellään aika, jonka välein viesti lähetetään väylälle sekä sallittu viive, joka kertoo maksimiviiveen, joka viestille hyväksytään. Vaikka kyseinen suorituskykytesti on alun perin tarkoitettu autoteollisuuden tarpeisiin, voidaan testin määrittelemien viestien ja viiveiden avulla simuloida oikean CAN-järjestelmän kuormia, prioriteetteja sekä viestejä. Kindell et al. [35, s. 5–7] esittävät SAE-suorituskykymittauksessa käytettävät viestitunnisteet, viesteille määritellyt lähetysaikavälit sekä sallitut viiveet. Tutkimuksessa esitetään myös laskelmia SAE-suorituskykymittauksen aiheuttamista väyläkuormista ja millä ehdoilla viestien lähetys toimii siten, että sallittua viivettä ei ylitetä.

SAE-suorituskykymittaus valittiin käytettäväksi lopullisen CANopen-siltatoteutuksen reititysviiveiden määrityksessä. Testiympäristöksi valittiin luvun 6.3 kaltainen järjestely, mutta nyt viestien lähetys tapahtuu kahteen suuntaan, kuten kuvassa 8.1 on esitetty. Viiveiden laskenta tapahtuu luvussa 6.3 esitettyjen periaatteiden mukaan. Testissä käytettiin baudinopeuksina 250, 500 ja 1000 kilobaudia. Testistä jätettiin pois 125 kilobaudin väylänopeus, koska Kindell et al. [35, s. 8–9] esittävät, että SAE-suorituskykytestistä aiheutuva väyläkuorma olisi 125 kilobaudin väylänopeudella yli 125 % eli huomattavasti yli maksimiväyläkuorman. Näin ollen on ymmärrettävää, että SAE-suorituskykytestiä käytettäessä 125 kilobaudin väylänopeudella ei voida taata, että kaikki viestit voidaan reitittää sallitussa ajassa.



Kuva 8.1. *SAE-suorituskykytestijärjestely.*

Testissä päädyttiin käyttämään SAE-suorituskykymittauksessa määriteltyjä 53 eri viestitunnisteen omaavaa viestiä sekä niille määriteltyjä sallittuja viiveitä. Viestit käyttivät CAN-viestitunnisteita 0x01-0x35. Koska kyseiset viestitunnisteet ovat kaikki korkean prioriteetin viestejä eli ne reititetään kaikki ajuritason kautta, päätettiin mukaan lisätä vielä seitsemän viestitunnistetta, jotka kuuluvut palveluobjektien viestitunnisteavaruuteen: 0x581, 0x582, 0x583, 0x584, 0x601, 0x602 sekä 0x603. Näin testiin saadaan mukaan myös sovellustason kautta reititettäviä viestejä. Viestit jaettiin siten, että kuvassa esitetyn Kvaser USB-CAN1-adapterin kautta lähetetään viestit, joilla on pariton viestitunniste ja Kvaser USB-CAN2-adapterin kautta lähetetään viestit, joilla on parillinen viestitunniste. Testeissä käytetyt viestitunnisteet, viesteille määritellyt lähetysaikavälit, sallitut viiveet sekä tulokset on esitetty liitteissä 2, 3 ja 4.

Tutkimalla liitteissä 2, 3 ja 4 esitettyjen taulukoiden maksimiviiveiden sarakkeita, huomataan, että korkean prioriteetin viestien (viestitunnisteet 0x01-0x35) maksimiviiveet pysyivät 500 ja 1000 kilobaudin väylänopeuksilla kaikissa tapauksissa alle yhden millisekunnin. 250 kilobaudin väylänopeudella korkean prioriteetin viestien maksimiviiveet olivat keskimäärin suurempia, mutta ne pysyivät kuitenkin alle kolmessa millisekunnissa. Sovellustason kautta reititettävien matalan prioriteetin palveluobjektiviestien maksimiviiveet olivat korkean prioriteetin viestejä suurempia, kuten luvussa 6.3.3 esitettyjen mittaustulosten pohjaltakin voidaan olettaa. Tutkimalla liitteiden 2, 3 ja 4 esittämien taulukoiden viimeisiä sarakkeita, jotka kertovat, kuinka paljon maksimiviive voisi kasvaa, jotta se saavuttaisi sallitun viiveen, voidaan huomata, että kaikki sarakkeissa esiintyvät arvot ovat positiivisia. Tämä tarkoittaa, että mikään viesteistä ei ylittänyt asetettuja sallittuja viiveitä valituilla väylänopeuksilla. Taulukkoon 8.1 on kerätty liitteiden 2, 3 ja 4 taulukoista eri baudinopeuksilta ne korkean ja matalan prioriteetin viestit, joiden maksimiviiveet olivat suhteellisesti lähimpänä sallittuja viiveitä. Näiden avulla voidaan määrittää, kuinka paljon tietyllä baudinopeudella voivat korkean ja matalan prioriteetin viestien reititysviiveet kasvaa siten, että ne kuitenkin pysyvät sallittujen rajojen sisällä.

Taulukko 8.1. *Lähimpänä sallittuja viiveitä olevat viestit eri väylänopeuksilla.*

Väylänopeus (kilobaudia)	Viestitunniste (hex)	Sallittu viive (us)	Max viive (us)	(Sallittu-Max)/Max (us)
250	2B	5000	2346	1,13
250	581	50000	33589	0,49
500	2A	5000	962	4,20
500	584	50000	36065	0,39
1000	31	5000	556	7,99
1000	582	50000	29258	0,71

Taulukosta huomataan, että 250 kilobaudin väylänopeudella korkean prioriteetin viestien reititysviiveet saisivat kasvaa maksimissaan 113 % ja matalan prioriteetin viestien reititysviiveet 49 % ennen kuin viestit saavuttaisivat sallitun viiveen. 500 kilobaudin väylänopeudella vastaavat lukemat ovat 420 % ja 39 % sekä 1000 kilobaudin nopeudella 799 % ja 71 %. Näin ollen voidaan todeta, että reititys on saatu optimoitua hyvälle tasolle ja reititykselle asetetut tavoitteet saavutetaan valituilla suunnitteluratkaisuilla. Ekiz et al. [34, s. 187] esittävät tutkimuksessaan samankaltaisia tuloksia kahteen segmenttiin jaetulle siltatoteutukselle.

9 YHTEENVETO

Tämän työn tavoitteena oli tutkia mahdollisuuksia liittää etähallintalaite osaksi CANopen-järjestelmää. Vaatimuksina olivat, että etähallintalaitteella tulee pystyä kuuntelemaan haluttuja prosessisignaaleita ja hätäviestejä sekä suorittamaan palveluobjektioperaatioita, joiden avulla järjestelmää voidaan etäkonfiguroida. Lisäksi menetelmälle asetettiin vaatimuksiksi, että etähallintalaitteen tulee olla yleiskäyttöinen ja etähallinnan liittäminen osaksi CANopen-järjestelmää ei saa aiheuttaa muutoksia väylän alkuperäisiin laitteisiin. Teoreettisen tarkastelun pohjalta oli tarkoituksena valita yksi menetelmä toteutettavaksi. Etähallintalaitteelle tuli määritellä ohjelmistorakenne, joka mahdollistaa valitun menetelmän toteuttamisen. Erityisesti kehityksen kohteena oli sovellustason CANopen-protokollamoduuli sekä CANopen-ohjelmointirajapinta.

Etähallintalaitteen liittämiseksi osaksi CANopen-järjestelmää valittiin yhteensä neljä erilaista menetelmää, jotka kaikki mahdollistavat prosessisignaalien ja hätäviestien kuuntelun sekä palveluobjektioperaariot. Menetelmistä laitetunnisteavaruuden jakaminen osiin sekä jaetun objektikanavan käyttö olivat potentiaalisia, mutta niiden käyttö vaatisi muutoksia alkuperäisen väylän laitteisiin. Niinpä kyseiset menetelmät jouduttiin sulkemaan pois. Nykyisen hallintalaitteen korvaaminen etähallinlaitteella puolestaan olisi hyvä ratkaisu, jos etähallintalaitteelta ei vaadittaisi geneerisyyttä. Lopulta menetelmäksi valittiin CANopen-silta, joka on CANopen-standardin ulkopuolinen menetelmä. Siltatoteutus vaatii etähallintalaitteelta kaksi CAN-liityntää, joista toiseen liitetään CANopen-hallintalaite ja toiseen loput väylän laitteet. Kaikki järjestelmät viestit reititetään etähallintalaitteen kautta. Silta mahdollistaa etähallintalaitteen liittämisen mihin tahansa CANopen-järjestelmään. Siltana toimiva etähallintalaite voi suorittaa palveluobjektioperaatioita orjalaitteille. Operaatiot tulee suorittaa siten, että alkuperäisen väylän toiminta ei häiriinny.

CANopen-silta mahdollistaa viestien reitityksen kahdella eri tasolla: sovellus- ja ajuritasolla. Suoritettujen mittausten perusteella todettiin, että ajuritasolla tulee suorittaa kaikkien järjestelmän kannalta kriittisten viestien reititys. Sovellustasolla voidaan reitittää viestit, joiden prioriteetti ei ole järjestelmän kannalta korkea. Mittaustulosten pohjalta lopullisessa sovelluksessa päädyttiin reitittämään kaikki muut paitsi palveluobjektiviestit ajuritasolla. Palveluobjektiviestit tulee reitittää sovellustason kautta, koska niiden käsittely vaatii sovellustason logiikkaa.

Lopullinen sovellus testattiin huolellisesti ja todettiin, että etähallintalaite havaitsee väylältä sekä hätäviestit että halutut prosessisignaalit tehtyjen suunnitteluratkaisujen avulla. Myös palveluobjektioperaatioiden suoritus testattiin useiden kaupallisten laitteiden avulla. Reitityksen suorituskykyä arvioitiin SAE-suorituskykymittauksella, joka

määrittelee 53 viestiä sekä viesteille lähetysvälit ja sallitut viiveet. Koska kyseiset 53 viestiä ovat kaikki korkean prioriteetin viestejä, päätettiin testiin lisätä seitsemän viestiä, joiden viestitunnisteet kuuluvat palveluobjekteille varattuun viestitunnisteavaruuteen. Näin kyseiset viestit reititetään sovellustason kautta. Testeissä väylänopeuksina käytettiin 250, 500 ja 1000 kilobaudin nopeuksia. Suorituskykytestin lopputuloksena todettiin, että reititys on saatu optimoitua riittävälle tasolle, koska yksikään viesti ei ylittänyt sille asetettua sallittua viivettä millään valituista väylänopeuksista. Näin ollen voidaan todeta, että suunnitteluratkaisut ja niiden avulla toteutettu sovellus ovat onnistuneet niille asetettujen tavoitteiden mukaisesti ja CANopen-siltatoteutus otetaan osaksi WRM-järjestelmää.

Tulevaisuudessa CANopen-siltaa voidaan kehittää edelleen. Siihen voidaan lisätä tuki esimerkiksi verkon hallintaisäntä toiminnallisuudelle, jolloin etähallintalaite voisi ohjata orjalaitteiden tiloja. Etähallintalaite voisi myös toimea esimerkiksi tahdistusviestien tuottaja. Etähallintalaitteeseen voitaisiin lisätä tuki orjalaitteena toimimiselle, jolloin etähallintalaite voisi tarjoa itse tekemiään mittauksia väylälle prosessisignaaliviesteinä. Lisäksi luvussa 6.3.3 esitettyjen reititysmittaustulosten perusteella WRM-etähallintalaitteen nykyisellä prosessorilla reitityksestä aiheutuvat prosessikuormat nousevat melko suuriksi. Tämä voidaan ottaa huomioon valitsemalla nopeampi prosessori tuleviin laiteversioihin.

LÄHTEET

[1] Wapice Oy:n kotisivu. [WWW]. [viitattu 13.3.2013]. Saatavissa: http://www.wapice.com/

[2] WRM247-laitteen kotisivu. [WWW]. [viitattu 13.3.2013]. Saatavissa: http://www.wrm.fi/

[3] CiA, CAN in Automation kotisivu. [WWW]. [viitattu 25.1.2013]. Saatavissa: http://can-cia.org/

[4] Pleiffer, O., Ayre, A. & Keydel, C. 2003. Embedded Networking with CAN and CANopen. San Clemente, RTC Books. 537 p.

[5] Saha, H. CANopen perusteet. FLUID Finland, 4(2006)1. s. 6-11.

[6] CiA WD-301. CANopen Application Layer and Communication Profile. Germany 2012, CAN in Automation (CiA). 199 p.

[7] CiA WD-306 Part 1. Electronic Data Sheet and Device Configuration File. Germany 2012, CAN in Automation (CiA). 28 p.

[8] Boterenbrood, H. CANopen: High Level Protocol for CAN-bus. 2005. Amsterdam, National Institute for Subatomic Physics. 23 p.

[9] CiA DSP-309 Part 1. Access from Other Networks: General Principles and Services. Germany 2006, CAN in Automation (CiA). 21 p.

[10] CiA DSP-413 Part 1. Device Profile for Truck Gateways: General Definitions. Germany 2011, CAN in Automation (CiA). 14 p.

[11] DeviceNET: Technical Overview. 2004. Michigan, USA, Open DeviceNET Vendor Association (ODVA). 8 p.

[12] Rinaldi, J. DeviceNET Introduction. [WWW]. Real Time Automation. 2009. [viitattu 16.3.2013]. Saatavissa: http://www.rtaautomation.com/devicenet/

[13] Edschberger, K. CAN-based Higher Layer Protocols and Profiles. [WWW]. IXXAT Inc. [viitattu 30.3.2013]. Saatavissa: http://www.ixxat.com/article_can_based_higher_layer_protocols_en.html

[14] Fredrikson, L. & Lennartsson, K. SDS, DeviceNET and CAN Kingdom. [WWW]. Kvaser AB. [viitattu: 25.4.2013]. Saatavissa: http://www.kvaser.com/zh/about-can/higher-layer-protocols/59.html

[15] Lian, F., Moyne J. & Tilbury, D. Performance Evaluation of Control Networks: Ethernet, ControlNet and DeviceNet. IEEE Control Systems Magazine, 21(2001)1. pp. 66-83.

[16] Fredrikson, L. A CAN Kingdom Rev 3.01. 1995. Kinnahult, Sweden, Kvaser AB. 109 p.

[17] Smart Distributed System Application Layer Protocol Version 2.0. 1996. USA, Honeywell Inc. 97 p.

[18] SAE J1939 Introduction. [WWW]. IXXAT Inc. [viitattu 4.5.2013]. Saatavissa: http://www.ixxat.com/introduction_sae_j1939_en.html

[19] Tuisku, T. 2012. CAN-väylä: Raskaankaluston standardi SAE 1939. Opinnäytetyö. Seinäjoki. Seinäjoen ammattikorkeakoulu, tietotekniikan koulutusohjelma. 33 s.

[20] Introduction to SAE J1939. [WWW]. Kvaser AB. [viitattu 4.5.2013]. Saatavissa: http://www.kvaser.com/zh/about-can/higher-layer-protocols/36.html

[21] Otten, K. & Gubel, C. J1939 C Library for CAN-Enabled PICmicro Microcontrollers. 2004. Microchip Technology Inc. 28 p.

[22] Lehtimäki, M. & Nevanperä, H. 2010. Isobus-järjestelmä työkoneohjauksessa. Opinnäytetyö. Seinäjoki. Seinäjoen ammattikorkeakoulu, auto- ja kuljetustekniikan koulutusohjelma. 51 s.

[23] Luft, L., Morschhauser, D. & Spitzer, S. NMEA 2000: Past, Present and Future. 2009. IMEA, International Marine Electronics Association. 25 p.

[24] CiA DSP-302 Part 5. SDO Manager. Germany 2009, CAN in Automation (CiA). 25 p.

[25] Ekiz, H., Kutlu, A. & Powner, E.T. Design and Implementation of a CAN/CAN Bridge. Parallel Architectures, Algorithms and Networks Second International Symposium Conference, Beijing, China, 12-14 June, 1996. 1996, IEEE. pp. 507-513.

[26] Saha, H. Active High-Speed CAN HUB, 11th international CAN Conference Part 2, Stockholm, Sweden, 2006. Germany, CAN in Automation, 2006. pp. 8-14.

[27] Saha, H. Comparison of System Level Networking Solutions with High-Speed CAN Networks. 9th international CAN Conference Part 9, Munich, Germany, 2003. Germany, CAN in Automation, 2003. pp. 1-8.

[28] Bäck, B., Nylund, P., Saha, H. & Wikman, M. An Improved CAN-Switch with CANopen-Management Interface. 12th international CAN Conference Part 2, Barcelona, Spain, 2012. Germany, CAN in Automation, 2012. pp. 8-15.

[29] Saha, H. Multilevel CANopen Networks. 10th international CAN Conference Part 7, Rome, Italy, 2005. Germany, CAN in Automation, 2005. pp. 1-6.

[30] Mitä eroa on keskittimellä, kytkimellä, reitittimellä ja tukiasemalla?. [WWW]. Microsoft Oy. [viitattu: 3.5.2013]. Saatavissa: http://windows.microsoft.com/fi-fi/windows-vista/how-do-hubs-switches-routers-and-access-points-differ

[31] Junnila, S., Pajula R., Shroff, M., Siuruainen, S., Kwitek, M. & Tuominen, P. Design of High-Performance CAN Driver Architecture for Embedded Linux. 13th international CAN Conference Part 5, Hamback Castle, Germany, March, 2012. Germany, CAN in Automation, 2012. pp. 1-9.

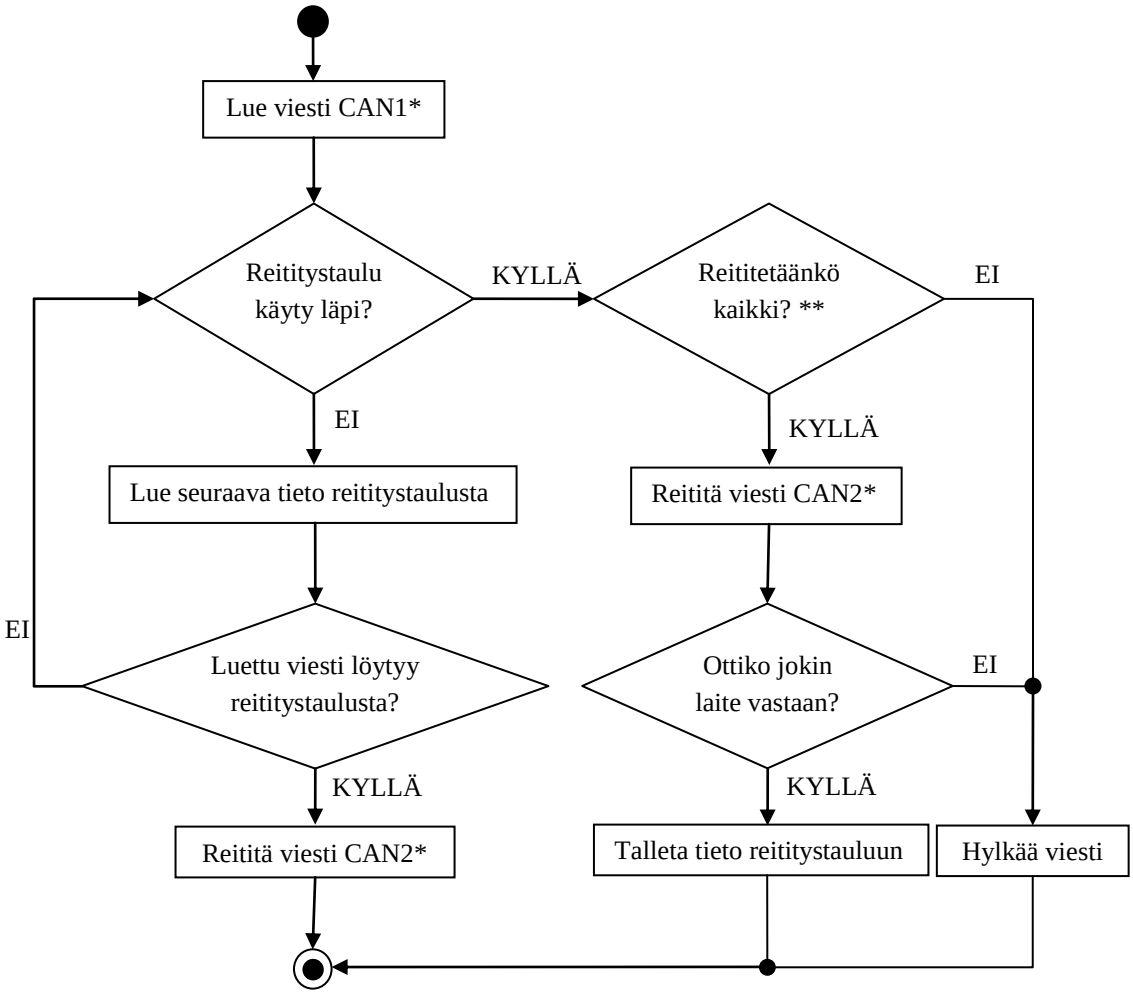
[32] Douglas, P. 2003. Real-Time Design Patterns: Robust Scalable Architecture for Real-Time Systems. Boston, USA, Pearson Education Inc. 477 p.

[33] Linux ioctl Command Man Page. [WWW]. [viitattu 23.4.2013]. Saatavissa: http://linux.die.net/man/2/ioctl

[34] Ekiz, H., Kutlu, A. & Powner, E.T. Implementation of CAN/CAN Bridges in Distributed Environments and Performance Analysis of Bridged CAN Systems Using SAE Benchmark. Engineering New Century Conference, Blacksburg, Virginia, USA, 12-14 April, 1997. 1997, IEEE. pp. 185-187.

[35] Tindell, K. & Burns, Guaranteeing Message Latencies on Control Area Network (CAN). 1st international CAN Conference, Germany, September, 1994. Germany 1994, CAN in Automation. pp. 1-11.

LIITE 1: CAN-SILTATOTEUTUKSEN ALGORITMI [25]



*Vaihtoehtoisesti CAN1 ja CAN2 voivat olla toisinpäin

** Kun silta on käynnistynyt, reititetään hetken kaikki viestit, koska reititystauluja vielä muodostetaan

LIITE 2: SAE-SUORITUSKYKYTESTIN VIESTIT JA TULOKSET 250 KILOBAUDIN VÄYLÄNOPEUDELLA

		Viiveet				
Viestitunniste (hex)	Lähetysväli (us)	Sallittu (us)	Min (us)	Ka (us)	Max (us)	(Sallittu-Max)/Max (us)
1	100000	100000	394	480,52	1403	70,28
2	100000	100000	372	504,94	1120	88,29
3	1000000	1000000	409	492,11	980	1019,41
4	100000	100000	385	514,67	1177	83,96
5	1000000	1000000	411	500,68	973	1026,75
6	100000	100000	391	525,52	1347	73,24
7	5000	5000	348	527,83	2266	1,21
8	5000	5000	317	525,07	2208	1,26
9	5000	5000	375	545,59	2125	1,35
A	100000	100000	382	542,52	1614	60,96
B	5000	5000	358	568,93	2338	1,14
C	100000	100000	384	555,99	1917	51,16
D	1000000	1000000	429	594,88	2048	487,28
E	50000	5000	367	470,13	936	4,34
F	50000	20000	394	516,42	1284	14,58
10	50000	20000	380	476,02	938	20,32
11	50000	20000	416	548,24	1320	14,15
12	20000	20000	338	540,73	2205	8,07
13	50000	20000	423	572,91	1429	13,00
14	50000	20000	395	487,14	924	20,65
15	1000000	1000000	456	619,88	2079	480,00
16	50000	20000	380	497,51	990	19,20
17	50000	20000	412	587,09	1430	12,99
18	50000	20000	390	507,96	1271	14,74
19	50000	20000	379	591,68	1434	12,95
1A	50000	20000	411	517,92	1547	11,93
1B	50000	20000	389	591,33	1413	13,15
1C	50000	20000	415	530,22	1534	12,04
1D	10000	10000	384	563,72	2646	2,78
1E	10000	10000	327	503,48	1501	5,66
1F	50000	20000	412	614,90	1497	12,36
20	5000	5000	331	559,79	2215	1,26
21	1000000	1000000	462	672,70	2068	482,56
22	50000	20000	393	571,68	1434	12,95
23	50000	20000	433	641,16	1464	12,66

24	1000000	1000000	384	537,39	866	1153,73
25	50000	20000	411	668,17	1494	12,39
26	50000	20000	413	589,67	1430	12,99
27	50000	20000	386	666,72	1583	11,63
28	50000	20000	381	610,55	1568	11,76
29	50000	20000	376	671,58	1452	12,77
2A	5000	5000	361	586,37	2250	1,22
2B	5000	5000	394	637,40	2346	1,13
2C	50000	20000	382	651,50	1684	10,88
2D	50000	20000	398	690,47	1508	12,26
2E	50000	20000	429	671,92	1975	9,13
2F	50000	20000	374	704,29	1673	10,95
30	50000	20000	385	690,25	1966	9,17
31	5000	5000	366	682,61	2275	1,20
32	50000	20000	415	718,89	1967	9,17
33	50000	20000	409	742,58	1938	9,32
34	50000	20000	430	750,52	1949	9,26
35	50000	20000	428	780,76	1954	9,24
581	1000000	50000	852	4155,58	33589	0,49
582	1500000	50000	910	2651,65	10515	3,76
583	2000000	50000	846	3740,52	26509	0,89
584	1000000	50000	1216	3029,62	13647	2,66
601	1500000	50000	889	3256,91	27967	0,79
602	2000000	50000	885	3392,29	30416	0,64
603	2500000	50000	989	3836,46	29785	0,68

LIITE 3: SAE-SUORITUSKYKYTESTIN VIESTIT JA TULOKSET 500 KILOBAUDIN VÄYLÄNOPEUDELLA

		Viiveet				
Viestitunniste (hex)	Lähetysväli (us)	Sallittu (us)	Min (us)	Ka (us)	Max (us)	(Sallittu-Max)/Max (us)
1	100000	100000	140	301,26	585	169,94
2	100000	100000	139	297,98	617	161,07
3	1000000	1000000	160	291,36	355	2815,90
4	100000	100000	116	275,54	655	151,67
5	1000000	1000000	201	286,61	351	2848,00
6	100000	100000	230	315,96	663	149,83
7	5000	5000	128	313,30	839	4,96
8	5000	5000	117	296,91	826	5,05
9	5000	5000	170	313,19	830	5,02
A	100000	100000	235	333,14	682	145,63
B	5000	5000	182	329,55	846	4,91
C	100000	100000	228	346,32	711	139,65
D	1000000	1000000	274	355,18	594	1682,5
E	50000	5000	120	281,90	450	10,11
F	50000	20000	221	313,35	525	37,1
10	50000	20000	184	308,58	539	36,11
11	50000	20000	216	328,37	625	31,00
12	20000	20000	118	319,23	789	24,35
13	50000	20000	255	344,98	641	30,20
14	50000	20000	233	321,12	707	27,29
15	1000000	1000000	285	392,50	629	1588,83
16	50000	20000	241	332,78	713	27,05
17	50000	20000	257	362,04	741	25,99
18	50000	20000	222	344,48	723	26,66
19	50000	20000	182	378,23	725	26,59
1A	50000	20000	240	358,27	730	26,40
1B	50000	20000	208	395,09	758	25,39
1C	50000	20000	232	370,76	729	26,43
1D	10000	10000	197	350,34	757	12,21
1E	10000	10000	121	321,20	667	13,99
1F	50000	20000	233	426,08	766	25,11
20	5000	5000	183	326,22	838	4,97
21	1000000	1000000	281	442,48	729	1370,74
22	50000	20000	216	400,21	756	25,46
23	50000	20000	260	439,87	781	24,61
24	1000000	1000000	196	329,11	633	1578,78
25	50000	20000	198	447,46	748	25,74

26	50000	20000	207	409,26	882	21,68
27	50000	20000	272	456,56	763	25,21
28	50000	20000	205	415,87	958	19,88
29	50000	20000	213	467,17	766	25,11
2A	5000	5000	205	342,69	962	4,20
2B	5000	5000	238	375,17	849	4,89
2C	50000	20000	227	426,92	976	19,49
2D	50000	20000	221	486,16	762	25,25
2E	50000	20000	263	433,76	988	19,24
2F	50000	20000	245	497,34	763	25,21
30	50000	20000	236	438,03	983	19,35
31	5000	5000	191	395,12	890	4,62
32	50000	20000	267	443,61	963	19,77
33	50000	20000	252	517,11	795	24,16
34	50000	20000	266	449,33	959	19,86
35	50000	20000	263	528,28	827	23,18
581	1000000	50000	609	1763,26	11039	3,53
582	1500000	50000	659	1857,65	4199	10,91
583	2000000	50000	701	1958,18	14321	2,49
584	1000000	50000	679	3321,96	36065	0,39
601	1500000	50000	803	1854,32	4720	9,59
602	2000000	50000	712	1537,73	4971	9,06
603	2500000	50000	740	1952,39	6476	6,72

LIITE 4: SAE-SUORITUSKYKYTESTIN VIESTIT JA TULOKSET 1000 KILOBAUDIN VÄYLÄNOPEUDELLA

		Viiveet				
Viestitunniste (hex)	Lähetysväli (us)	Sallittu (us)	Min (us)	Ka (us)	Max (us)	(Sallittu-Max)/Max (us)
1	100000	100000	152	199,11	387	257,40
2	100000	100000	143	198,56	259	385,10
3	1000000	1000000	180	201,18	255	3920,57
4	100000	100000	121	198,44	250	399,00
5	1000000	1000000	161	199,24	244	4097,36
6	100000	100000	102	201,76	318	313,47
7	5000	5000	105	196,94	373	12,40
8	5000	5000	116	204,53	386	11,95
9	5000	5000	96	196,48	395	11,66
A	100000	100000	114	211,66	316	315,46
B	5000	5000	107	199,36	431	10,60
C	100000	100000	145	213,77	352	283,09
D	1000000	1000000	167	210,60	362	2761,43
E	50000	5000	121	202,40	357	13,01
F	50000	20000	150	205,14	373	52,62
10	50000	20000	122	206,43	374	52,48
11	50000	20000	129	209,26	379	51,77
12	20000	20000	106	195,68	429	45,62
13	50000	20000	120	213,59	422	46,39
14	50000	20000	130	212,29	407	48,14
15	1000000	1000000	164	213,18	381	2623,67
16	50000	20000	130	214,98	400	49,00
17	50000	20000	132	215,89	395	49,63
18	50000	20000	130	216,54	361	54,40
19	50000	20000	100	220,43	398	49,25
1A	50000	20000	117	218,27	380	51,63
1B	50000	20000	129	225,07	409	47,90
1C	50000	20000	136	220,87	403	48,63
1D	10000	10000	105	213,11	429	22,31
1E	10000	10000	102	204,82	423	22,64
1F	50000	20000	142	233,04	517	37,68
20	5000	5000	110	209,35	432	10,57
21	1000000	1000000	169	219,24	358	2792,30
22	50000	20000	113	221,72	402	48,75
23	50000	20000	148	236,43	497	39,24

24	1000000	1000000	133	213,31	276	3622,19
25	50000	20000	150	238,80	522	37,31
26	50000	20000	107	221,72	403	48,63
27	50000	20000	137	239,86	547	35,56
28	50000	20000	115	222,52	379	51,77
29	50000	20000	141	238,56	519	37,54
2A	5000	5000	60	213,44	438	10,42
2B	5000	5000	115	211,42	531	8,42
2C	50000	20000	136	221,51	420	46,62
2D	50000	20000	133	234,85	526	37,02
2E	50000	20000	103	219,93	432	45,30
2F	50000	20000	142	232,92	543	35,83
30	50000	20000	108	218,73	424	46,17
31	5000	5000	100	213,81	556	7,99
32	50000	20000	108	218,08	451	43,35
33	50000	20000	119	231,07	581	33,42
34	50000	20000	69	217,52	432	45,30
35	50000	20000	108	230,66	582	33,36
581	1000000	50000	543	2943,18	28960	0,73
582	1500000	50000	615	3636,14	29258	0,71
583	2000000	50000	499	2971,59	28175	0,77
584	1000000	50000	512	2272,58	7862	5,36
601	1500000	50000	673	1907,98	12564	2,98
602	2000000	50000	709	1951,54	8588	4,82
603	2500000	50000	690	1771,48	5595	7,94